UNIVERSITY OF Southampton
School of Electronics and Computer Science

Ontology Design Patterns

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Questions

- How can we represent an ordered list?
  - E.g. want to describe a bus route, how can we represent the sequence of stops?
- How can we add information to a relation (property)?
  - E.g. need to set a confidence value to the relation
- How do we represent lists of values?
  - E.g. a fixed list of airline models

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Topics

- N-ary relations
  - How can we say more about a relation instance?
- Classes as property values
  - What do we do if we need to use a Class as a property value?
- Value partitions and value sets
  - How do we represent a fixed list of values?

UNIVERSITY OF Southampton
School of Electronics and Computer Science

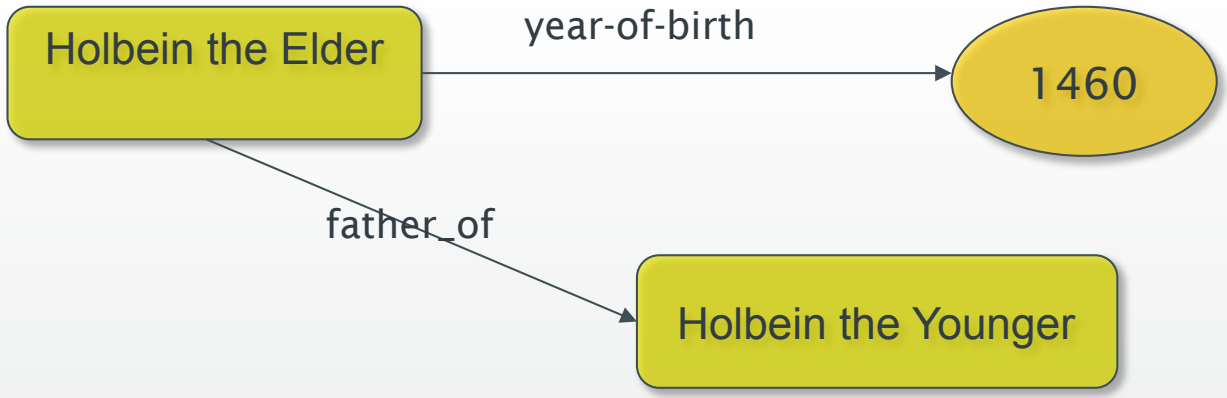
Topics

- N-ary relations
- Classes as property values
- Value partitions and Value sets

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Binary Relations

- In RDF and OWL, binary relations link two individuals, or an individual and a value

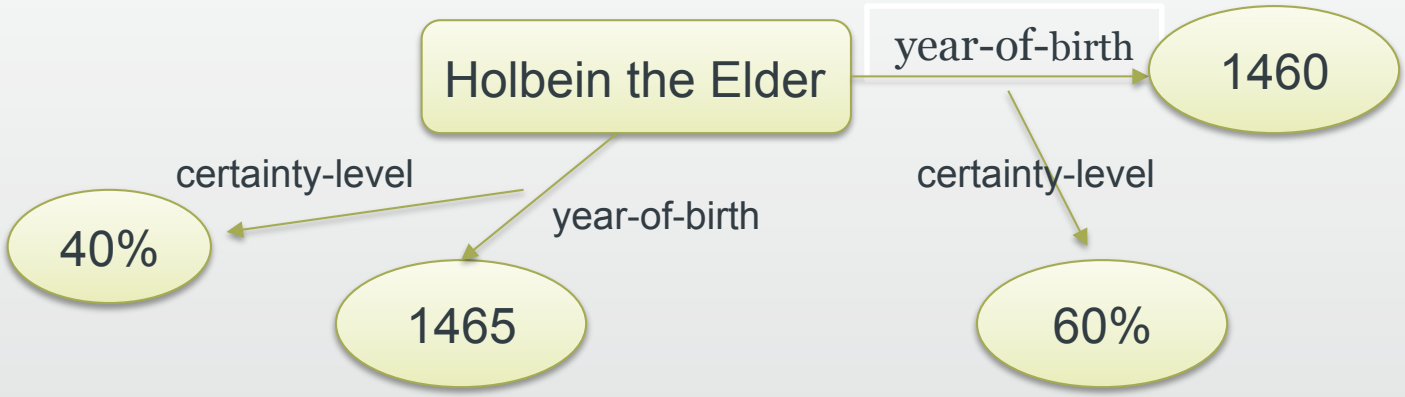


- The properties year-of-birth and father-of are binary relations

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Relations with additional info

- In some cases, we need to associate additional info with a binary relation
  - Eg certainty, strength, dates
- For example, Holbein the Elder's date of birth is unconfirmed
  - He was born in either 1460 or 1465
  - How can we represent this uncertainty?



N-ary Relations

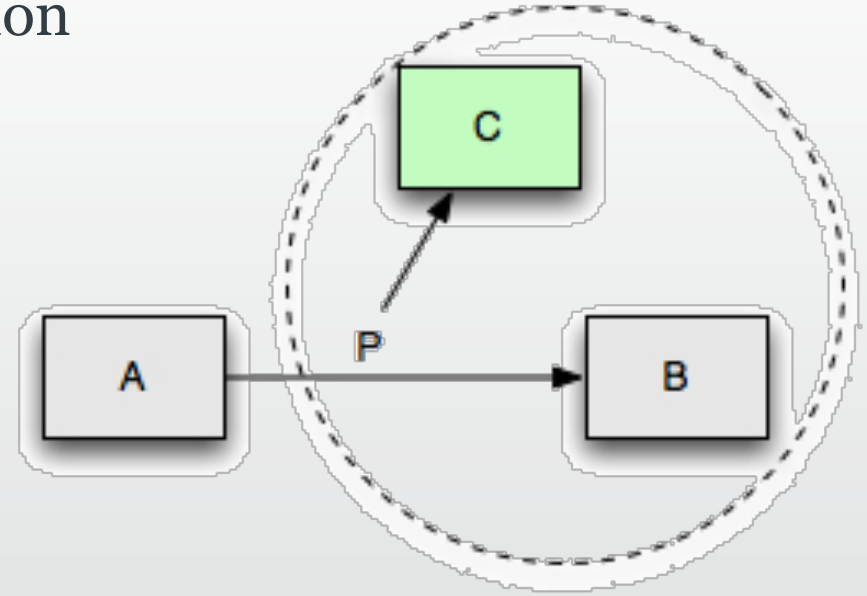
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- N-ary relations link an individual to more than a single individual or value
- Use cases:
  1. A relation needs additional info
     - eg a relation with a rating value
  2. Two binary relations are related to each other
     - eg body_temp (high, normal, low), and trend (rising, falling)
  3. A relation between several individuals
     - eg *someone* buys a *book* from a *bookstore*
  4. Linking from, or to, an ordered list of individuals
     - eg an airline flight visiting a sequence of airports
- Pattern 1: Creating a new class or relation
  - Use for cases 1, 2, and 3 above
- Pattern 2: Sequence of arguments
  - For case 4

N-ary relation - Pattern 1: Creating a new class or relation

UNIVERSITY OF Southampton
School of Electronics and Computer Science

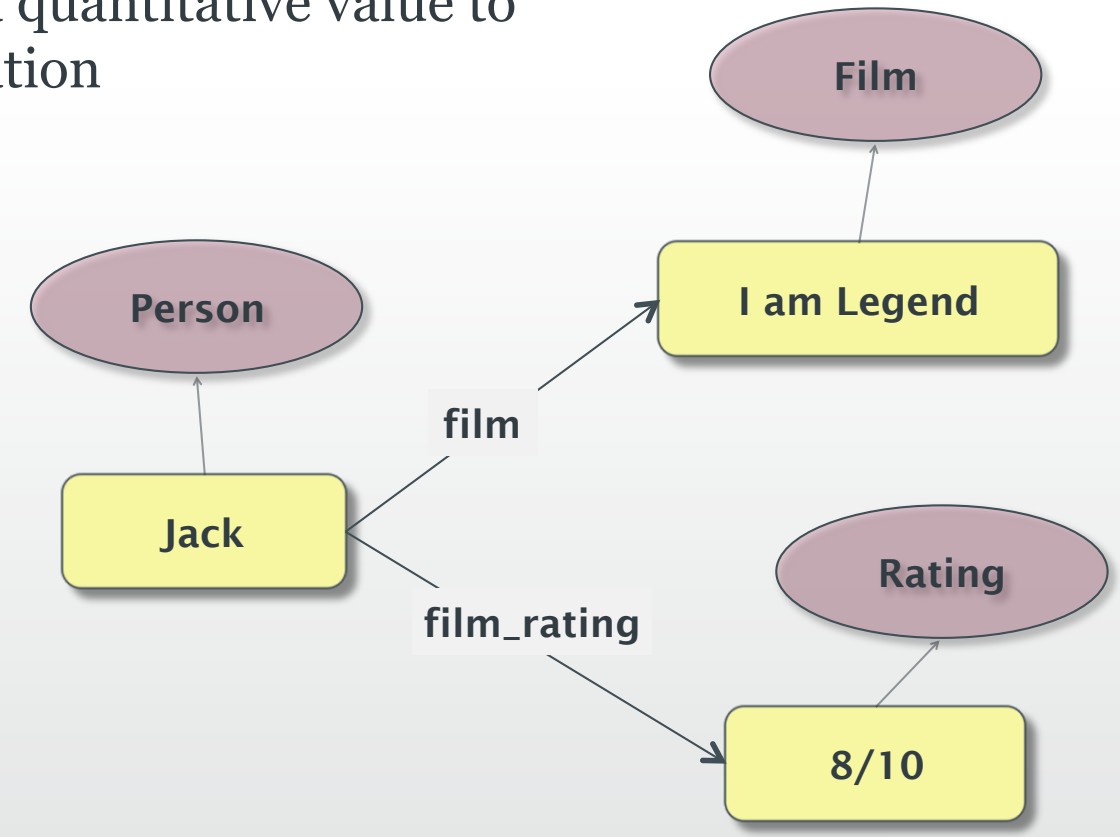
- To represent additional information about a relations:
  - We can create a new class to represent the relation
  - The individuals of this class are instances of the relation
  - This class can have additional properties to describe more information about the relation



Use case 1: additional information about a relation

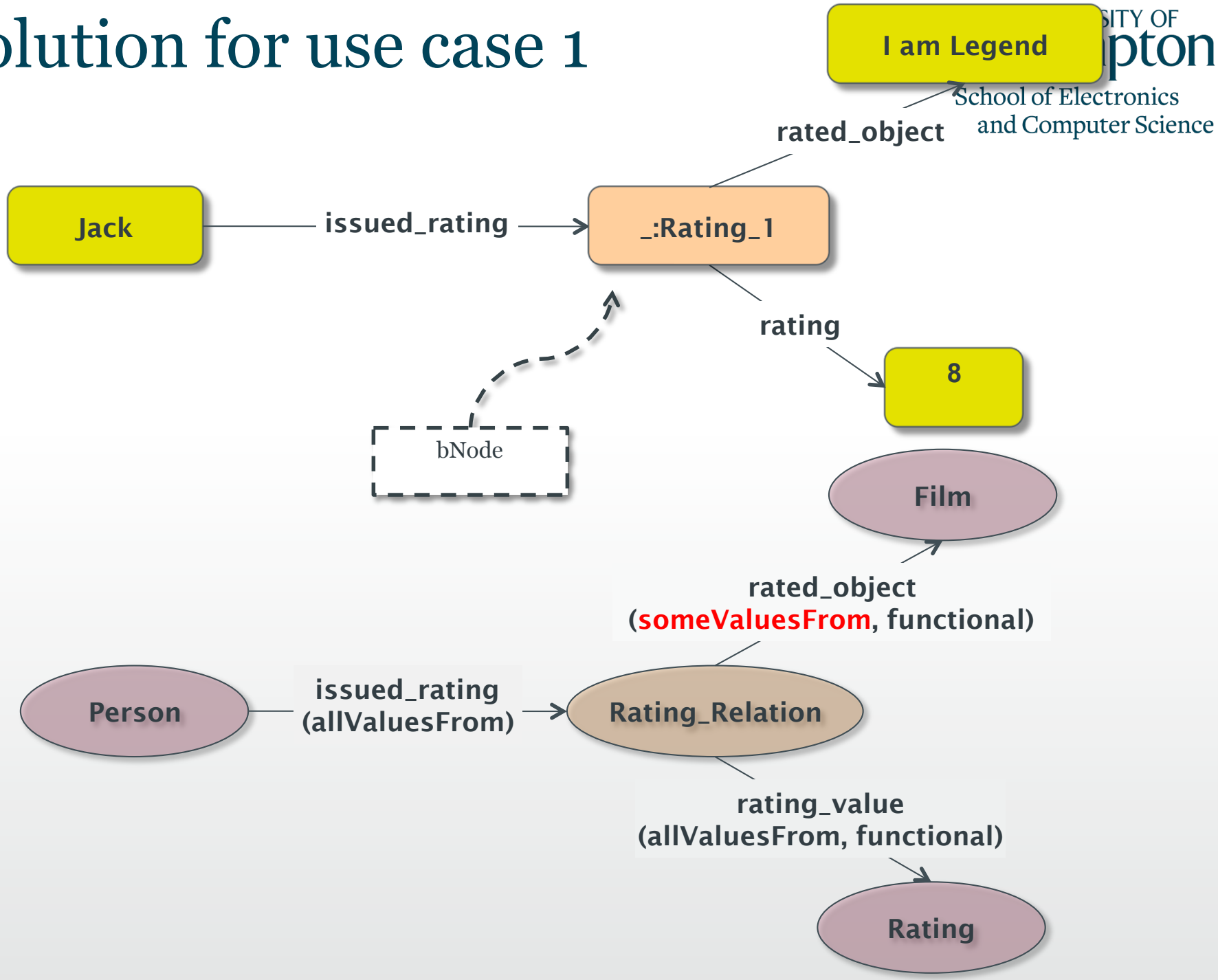
UNIVERSITY OF Southampton
School of Electronics and Computer Science

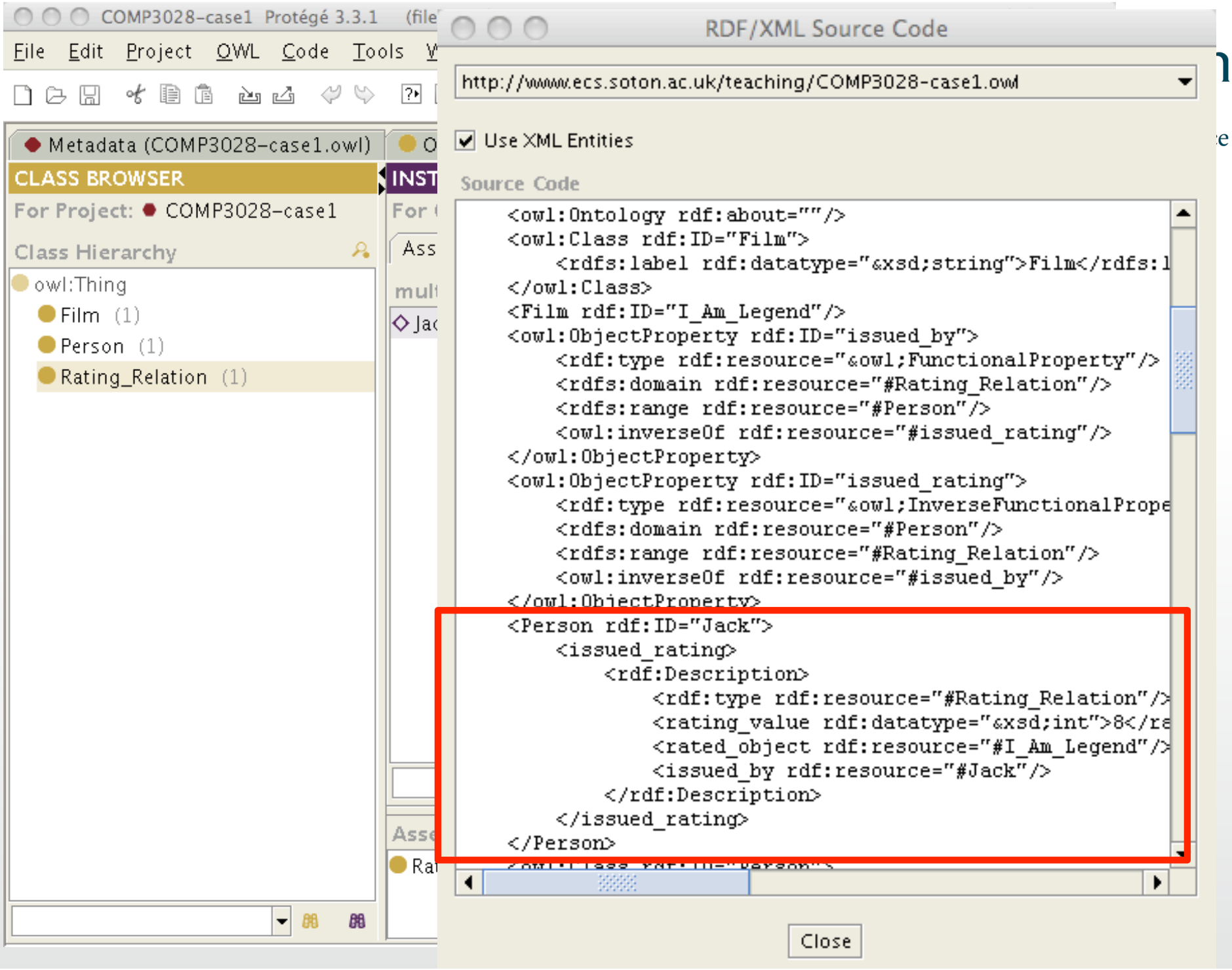
- Jack has given the film '*I Am Legend*' a rating of 8
- We need to represent a quantitative value to describe the rating relation
  - *What is wrong with this representation?*
    - *What will happen when Jack rates other films?*



Solution for use case 1

School of Electronics
and Computer Science

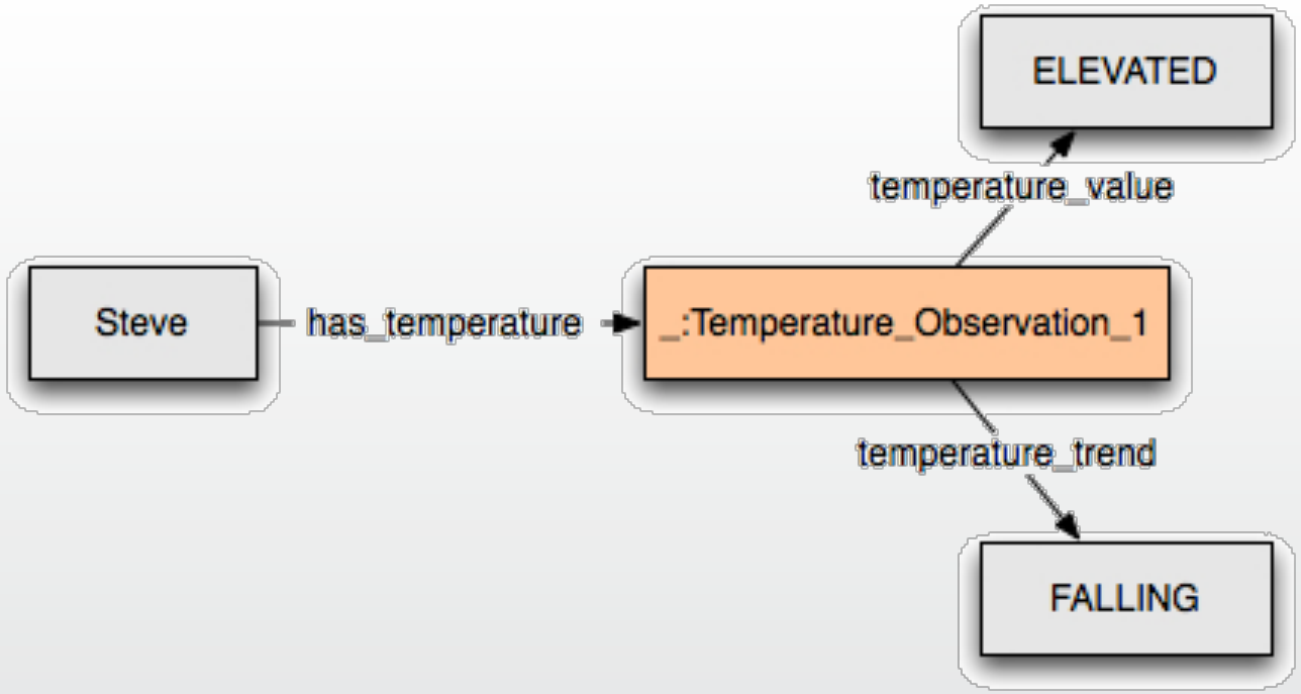



COMP3028-case1 Protégé 3.3.1
File Edit Project OWL Code Tools
Metadata (COMP3028-case1.owl)
CLASS BROWSER
For Project: COMP3028-case1
Class Hierarchy
owl:Thing
Film (1)
Person (1)
Rating_Relation (1)
RDF/XML Source Code
http://www.ecs.soton.ac.uk/teaching/COMP3028-case1.owl
Use XML Entities
Source Code
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Film">
<rdfs:label rdf:datatype="&xsd;string">Film</rdfs:l
</owl:Class>
<Film rdf:ID="I_Am_Legend"/>
<owl:ObjectProperty rdf:ID="issued_by">
<rdf:type rdf:resource="&owl;FunctionalProperty"/>
<rdfs:domain rdf:resource="#Rating_Relation"/>
<rdfs:range rdf:resource="#Person"/>
<owl:inverseOf rdf:resource="#issued_rating"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="issued_rating">
<rdf:type rdf:resource="&owl;InverseFunctionalPrope
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range rdf:resource="#Rating_Relation"/>
<owl:inverseOf rdf:resource="#issued_by"/>
</owl:ObjectProperty>
<Person rdf:ID="Jack">
<issued_rating>
<rdf:Description>
<rdf:type rdf:resource="#Rating_Relation"/>
<rating_value rdf:datatype="&xsd;int">8</ra
<rated_object rdf:resource="#I_Am_Legend"/>
<issued_by rdf:resource="#Jack"/>
</rdf:Description>
</issued_rating>
</Person>
Close

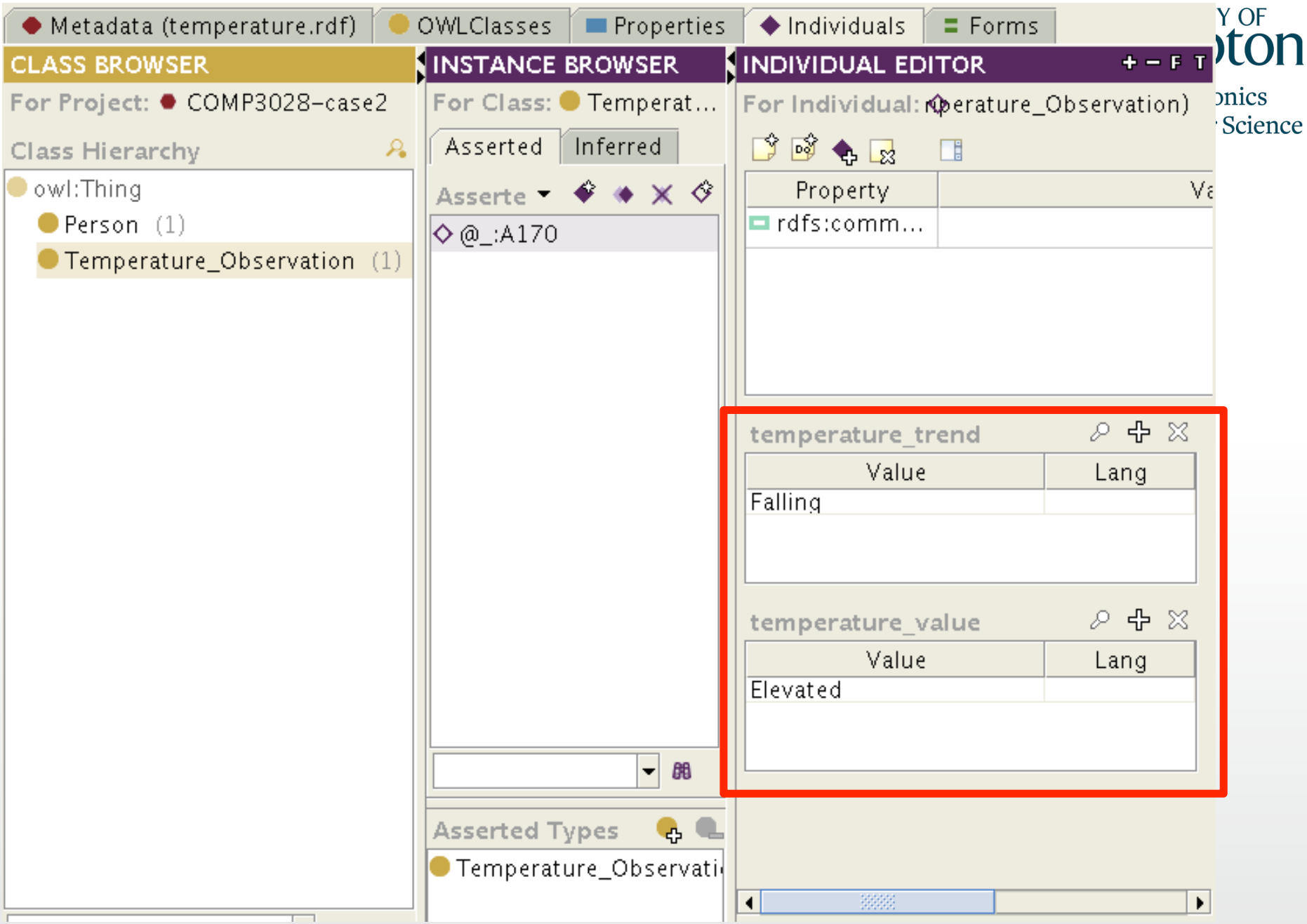
UNIVERSITY OF Southampton
School of Electronics and Computer Science

Use case 2: different aspects of the same relation

- *Steve has temperature, which is high, but falling*
- We need to represent different aspects of the temperature that Steve has



Source: W3C

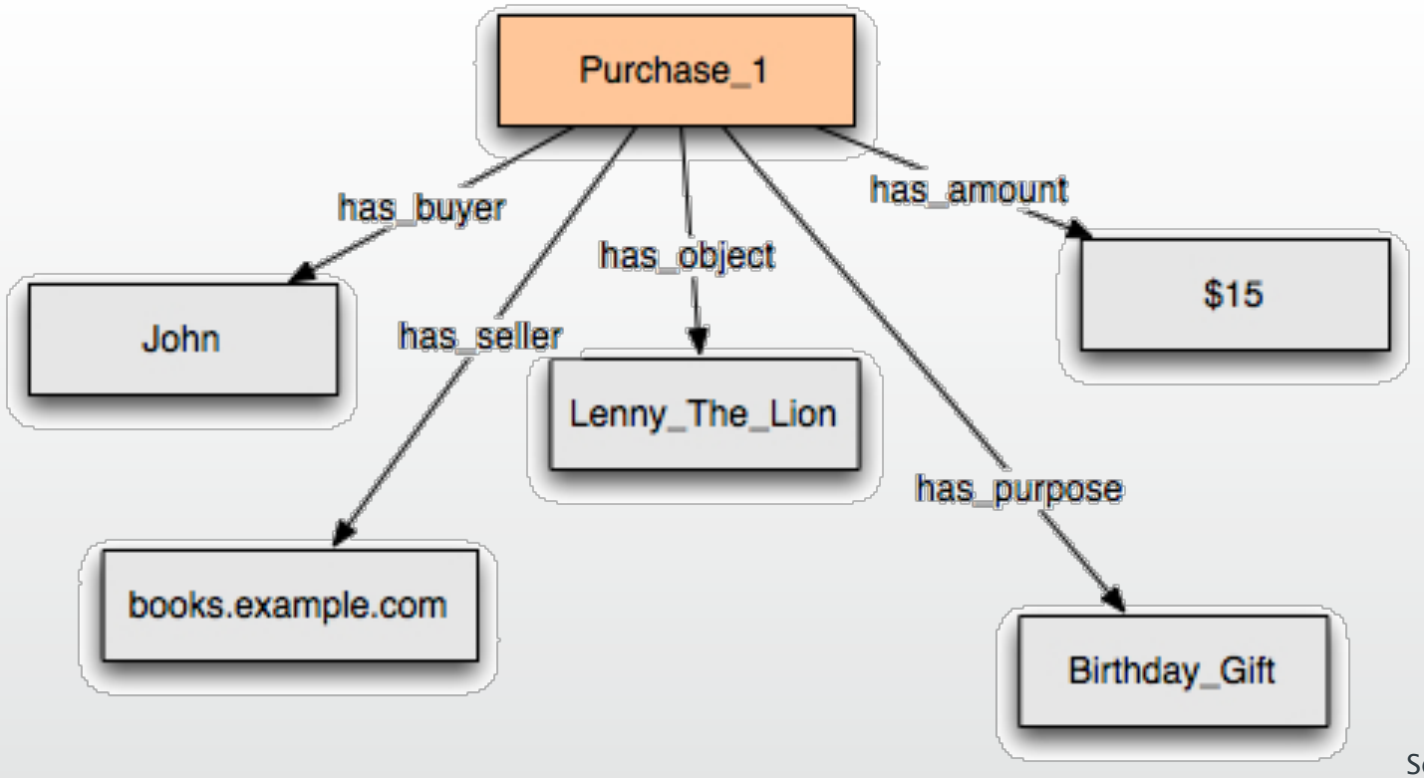


http://www.w3.org/TR/swbp-n-aryRelations/temperature.rdf

Use case 3: N-ary relation with no distinguished participant

UNIVERSITY OF Southampton
School of Electronics and Computer Science

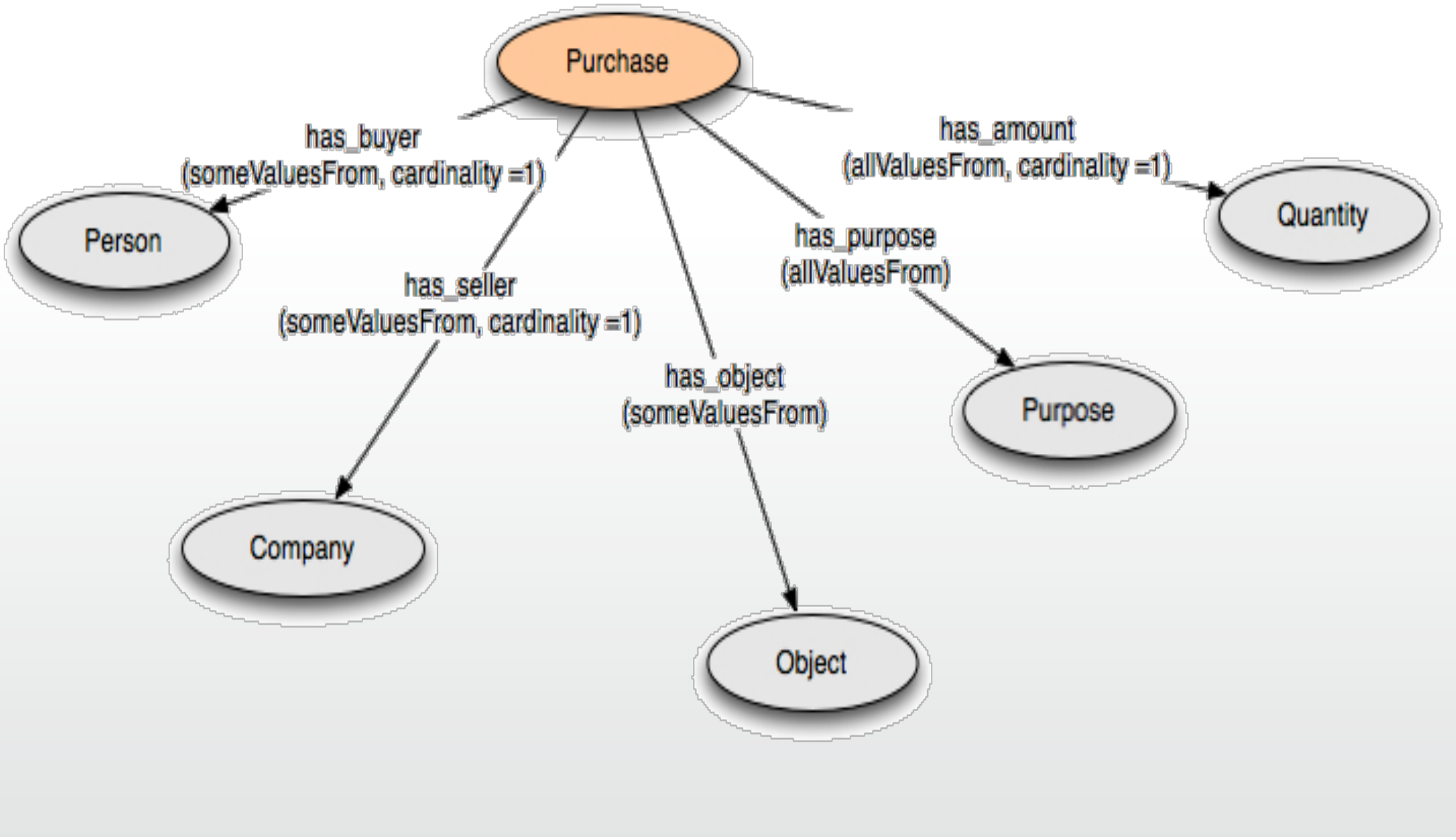
- *John buys a "Lenny the Lion" book from books.example.com for $15 as a birthday gift*
- No distinguished subject for the relation
  - i.e. no primary relation to convert into a Relation Class as in cases 1 and 2



Source: W3C

Solution for use case 3

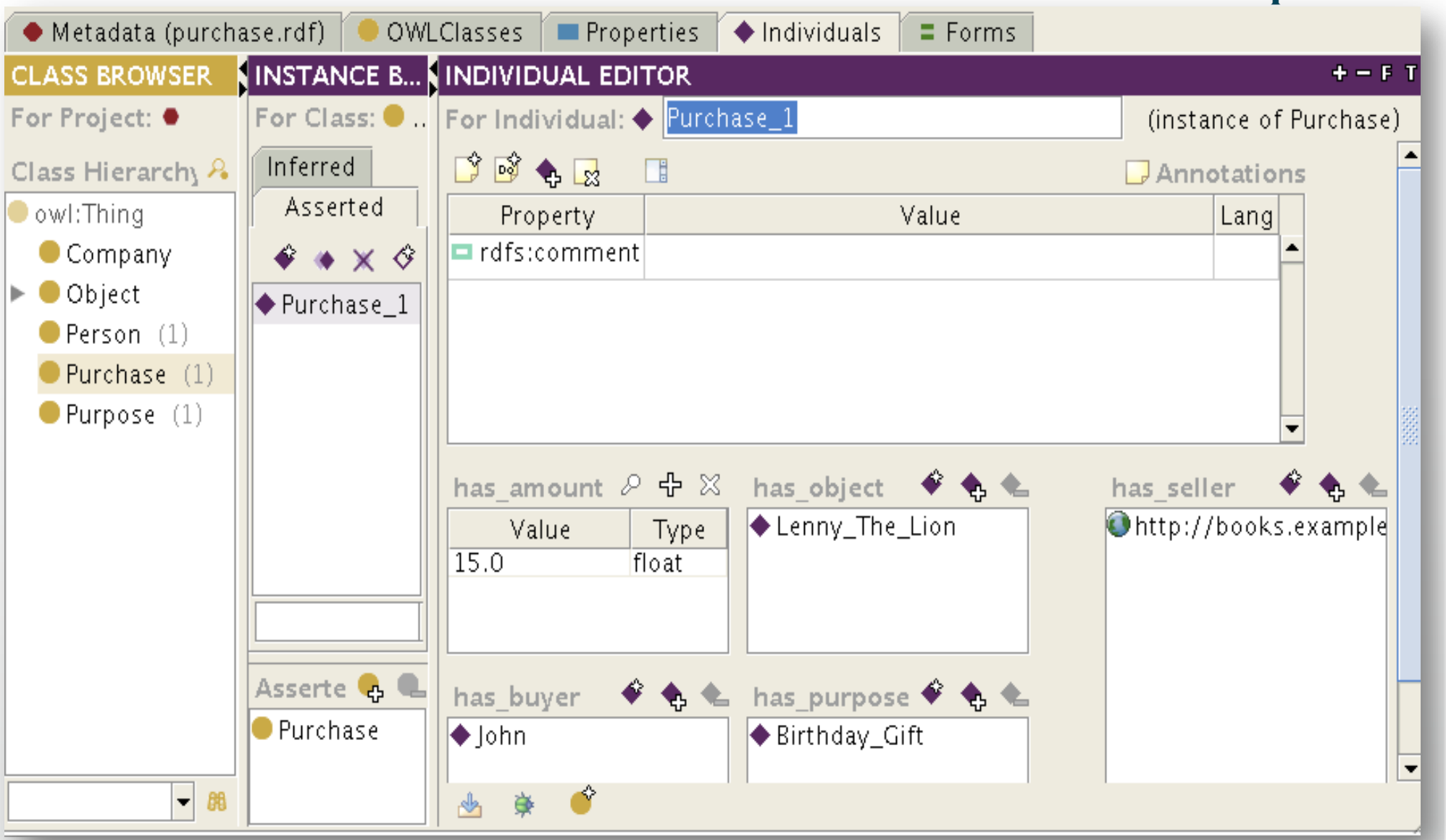
UNIVERSITY OF Southampton
School of Electronics and Computer Science



UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
<rdfs:Class rdf:ID="Purchase"/>
<rdfs:Class rdf:ID="Person"/>
<rdfs:Class rdf:ID="Company"/>
<rdf:Property rdf:ID="has_seller">
  <rdfs:range rdf:resource="#Company"/>
  <rdfs:domain rdf:resource="#Purchase"/>
</rdf:Property>
<rdf:Property rdf:ID="has_object">
  <rdfs:range rdf:resource="#Object"/>
  <rdfs:domain rdf:resource="#Purchase"/>
</rdf:Property>
<rdf:Property rdf:ID="has_buyer">
  <rdfs:domain rdf:resource="#Purchase"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property rdf:ID="has_purpose">
  <rdfs:range rdf:resource="#Purpose"/>
  <rdfs:domain rdf:resource="#Purchase"/>
</rdf:Property>
<rdf:Property rdf:ID="has_amount">
  <rdfs:domain rdf:resource="#Purchase"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</rdf:Property>
<Person rdf:ID="John"/>
<Purchase rdf:ID="Purchase_1">
  <has_amount rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >15.0</has_amount>
  <has_object>
    <Book rdf:ID="Lenny_The_Lion"/>
  </has_object>
  <has_buyer rdf:resource="#John"/>
  <has_seller rdf:resource="http://books.example.com"/>
  <has_purpose>
    <Purpose rdf:ID="Birthday_Gift"/>
  </has_purpose>
```

UNIVERSITY OF Southampton

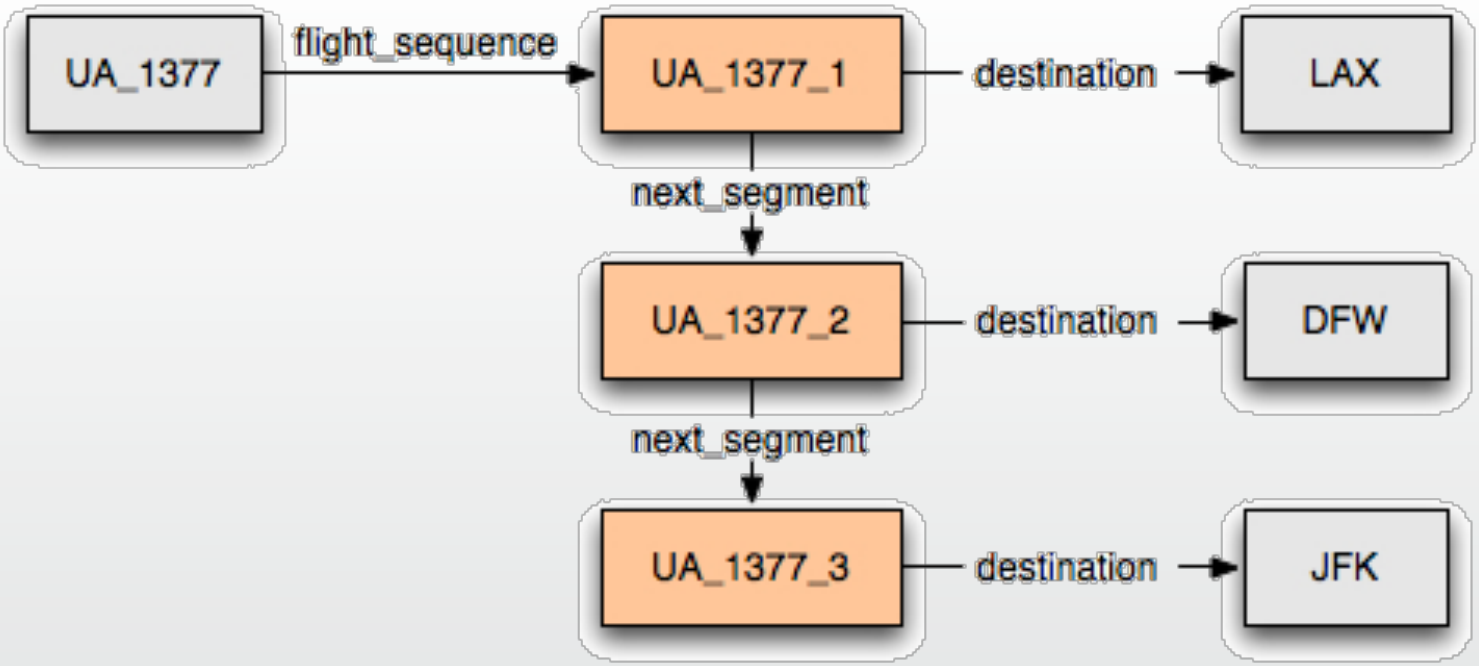


http://www.w3.org/TR/swbp-n-aryRelations/purchase.rdf

N-ary Relations - Pattern 2: Sequence of arguments

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- *United Airlines, flight 1377 visits the following airports: LAX, DFW, and JFK*
- For such an example, we need to represent a sequence of arguments

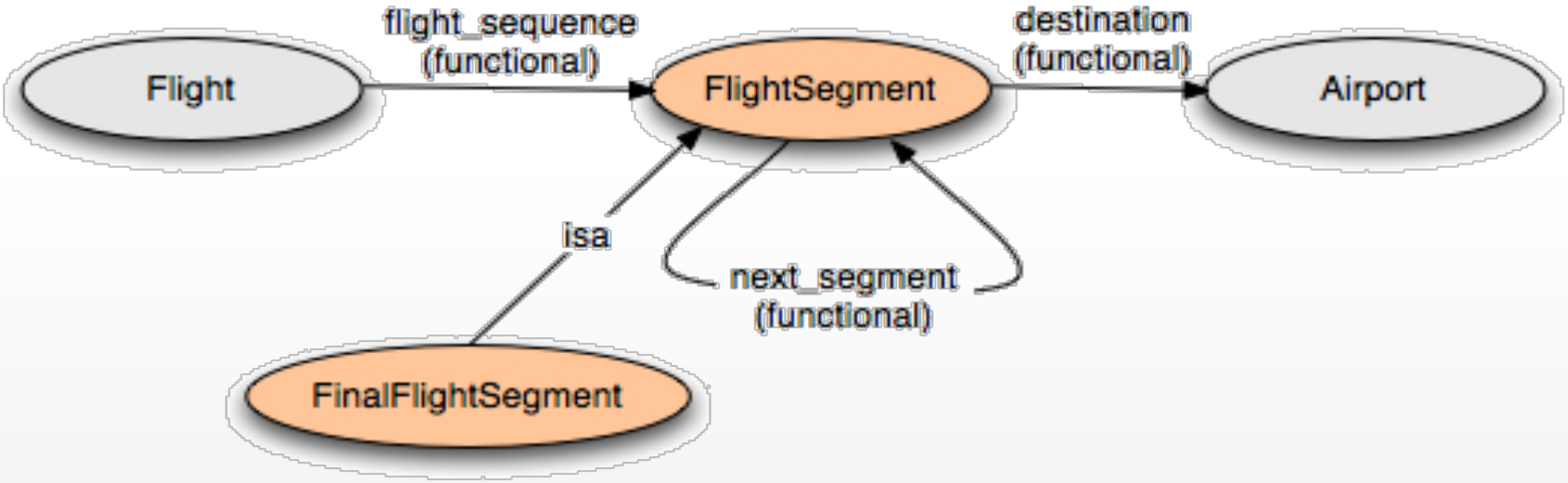


Source: W3C

N-ary Relations - Pattern 2: Sequence of arguments

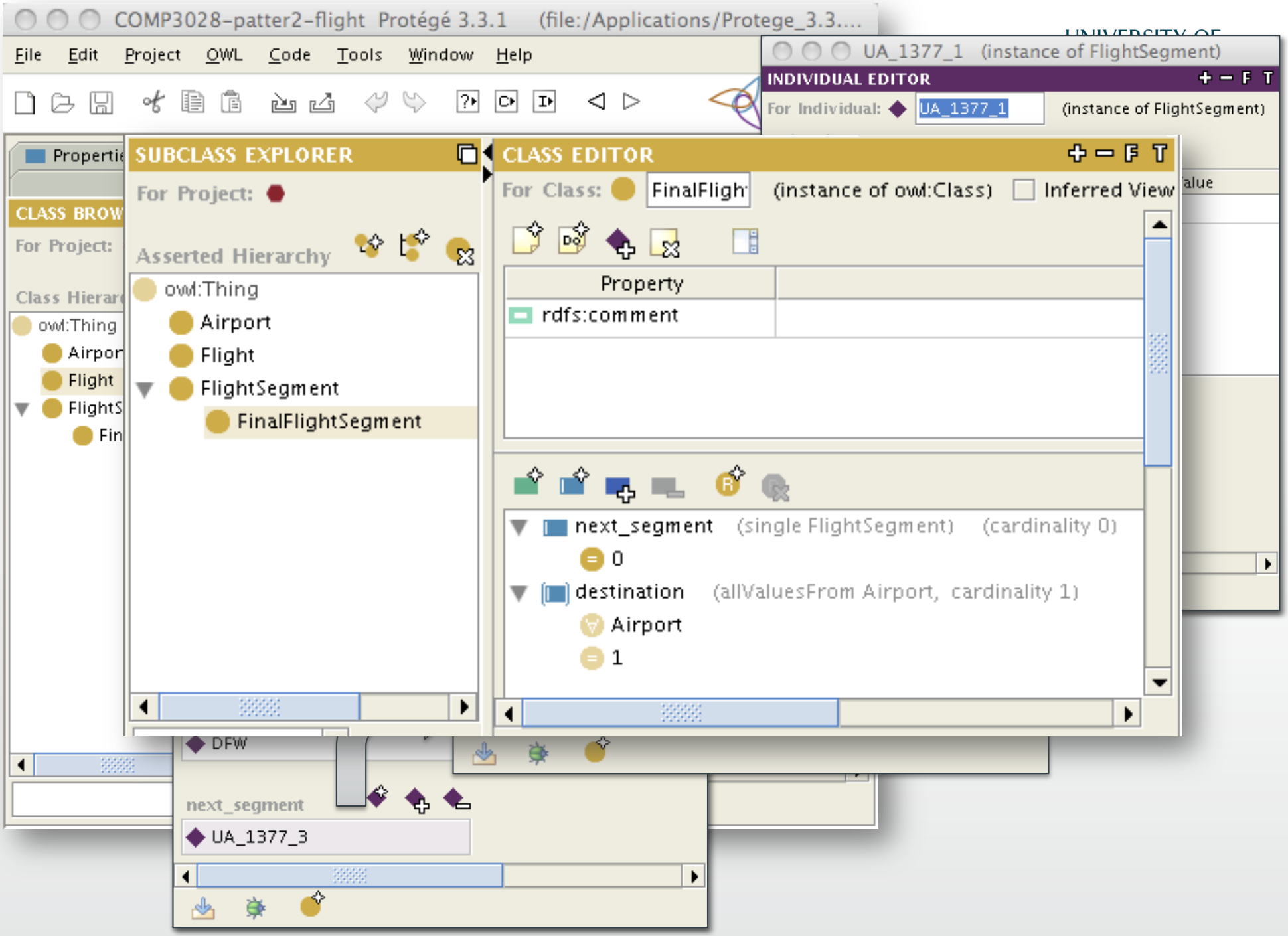
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- This is the OWL:Lite ontology to represent a sequence




```
:FinalFlightSegment a owl:Class ;
        rdfs:comment "The last flight segment has no next_segment";
        rdfs:subClassOf :FlightSegment ;
        rdfs:subClassOf
                [ a owl:Restriction ; owl:maxCardinality "0";
                    owl:onProperty :next_segment] .
```

Source: W3C



http://users.ecs.soton.ac.uk/ha/teaching/COMP3028/pattern2-flight-sequence.owl

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Topics

- N-ary relations
- Classes as property values
- Value partitions and Value sets

UNIVERSITY OF Southampton
School of Electronics and Computer Science

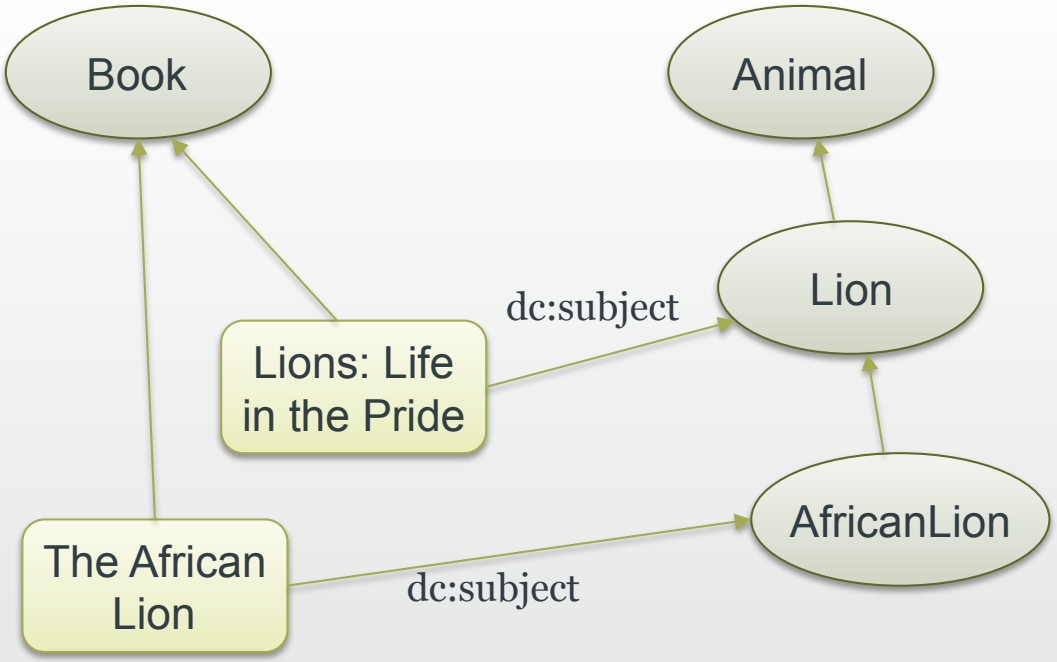
Classes as property values

- In some cases, it is convenient to put a class as a value of some property
- Classes can be property values in **RDFS** and **OWL Full**, with no restrictions
- In OWL DL and OWL Lite, classes cannot be property values
  - Because nothing can be both a class and an individual
  - Need to use alternative mechanisms

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Use case example

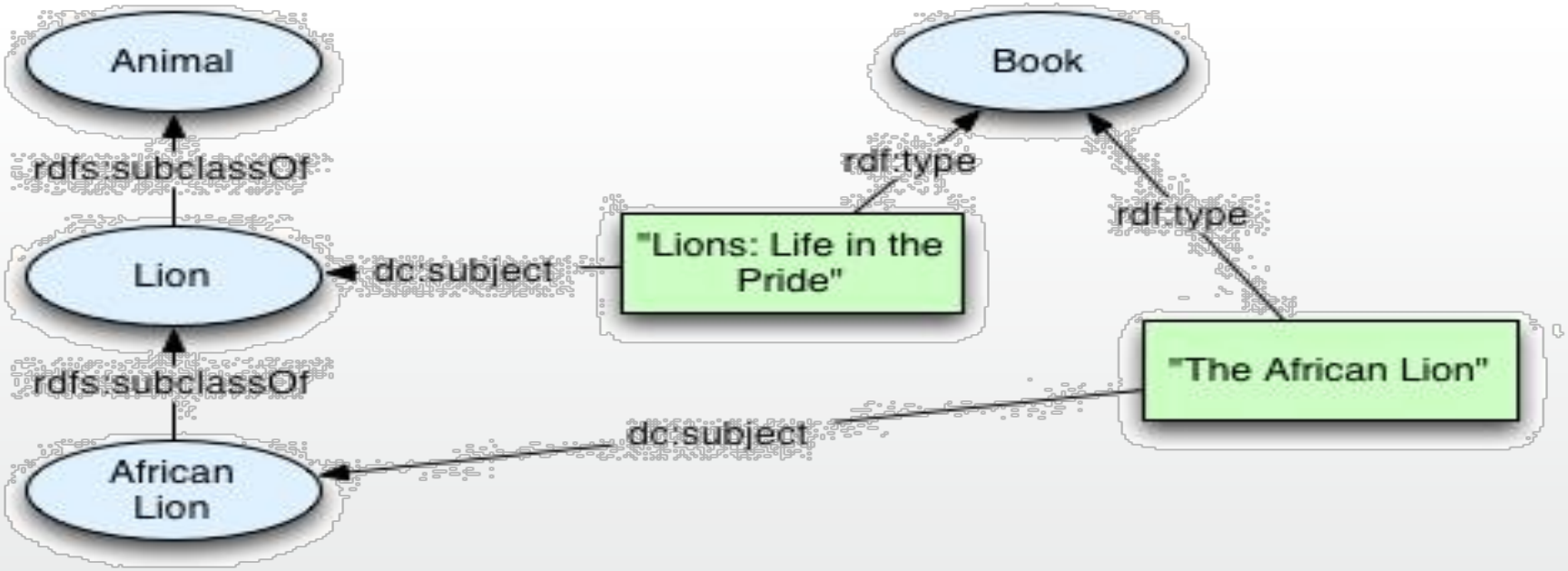
- Represent two books about lions, one is about the species of lion, and the other about the species of African lion
- Retrieve both books when asking for books about lions



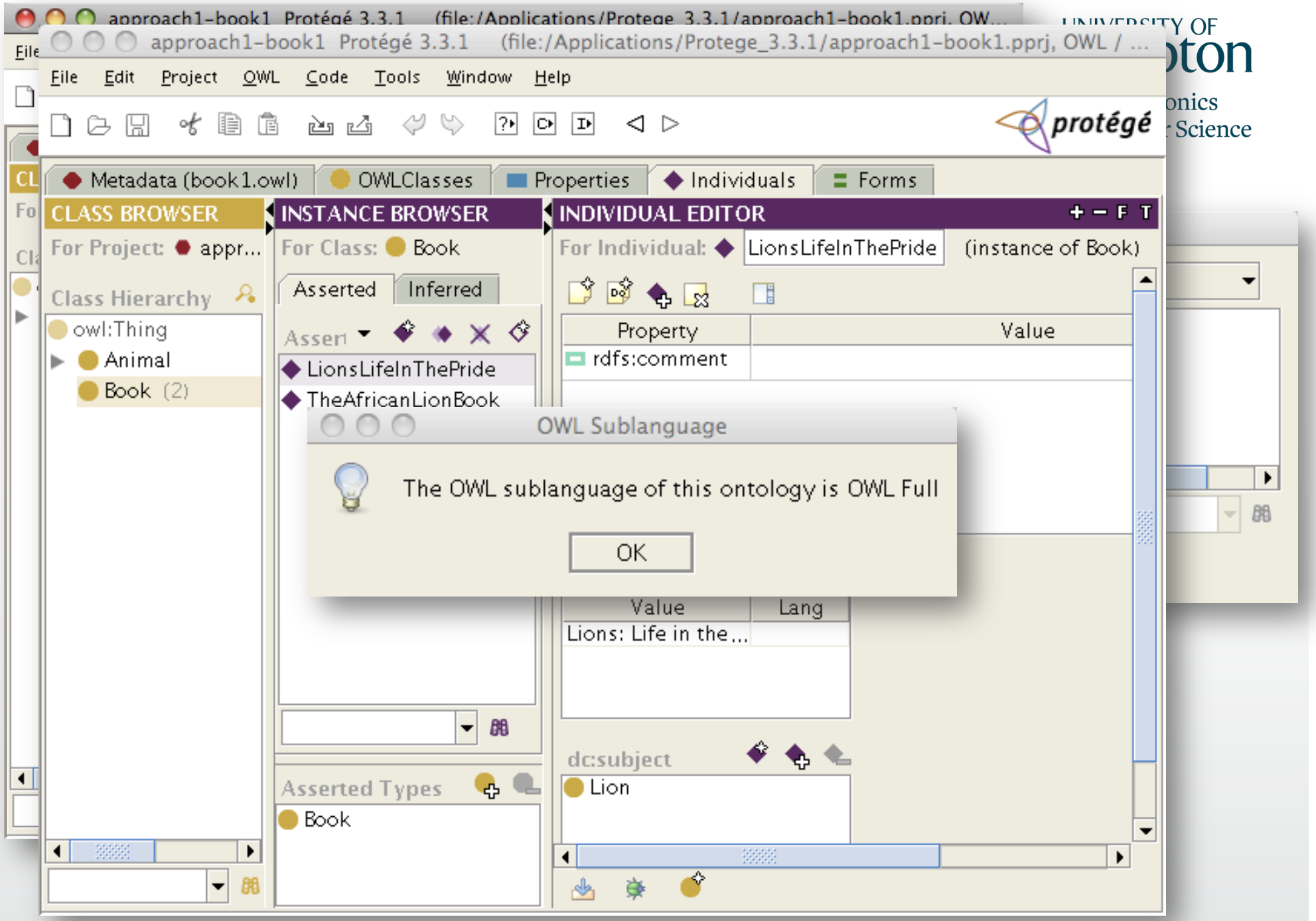
Approach 1:
Use classes directly as property values

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- The property dc:subject has the Animal classes as values



Source: W3C



http://users.ecs.soton.ac.uk/ha/teaching/COMP3028/approach1-book1.owl

SPARQL Query

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX dc:  <http://purl.org/dc/elements/1.1/>

PREFIX base:  <http://www.ecs.soton.ac.uk/teaching/COMP3028/
   book1.owl#>


SELECT ?book

WHERE { ?book dc:subject ?subject .

 ?subject rdfs:subClassOf base:Lion}
```

```
<http://www.ecs.soton.ac.uk/teaching/COMP3028/book1.owl#LionsLifeInThePrideBook>
<http://www.ecs.soton.ac.uk/teaching/COMP3028/book1.owl#AfricanLionBook>
```

UNIVERSITY OF Southampton
School of Electronics and Computer Science

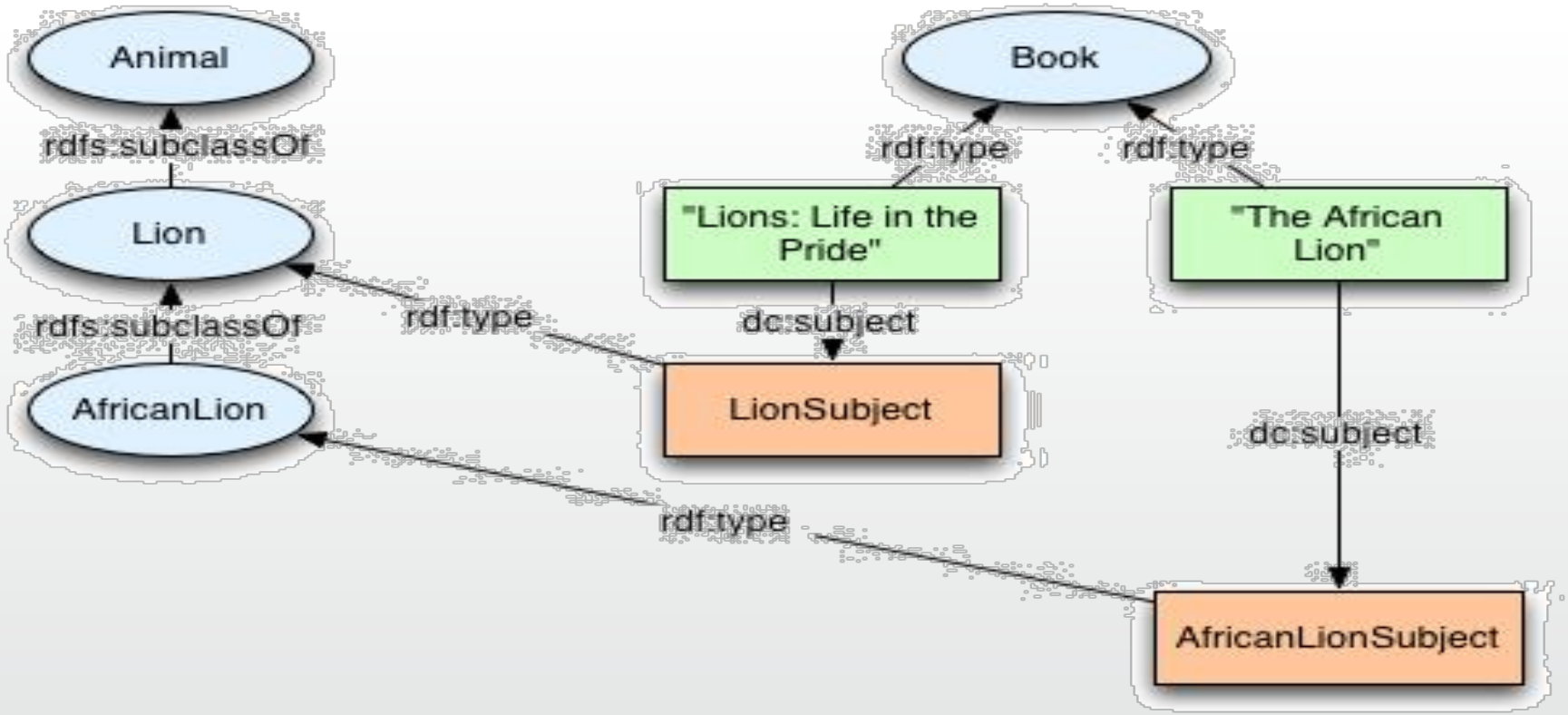
Notes on Approach 1

- This approach is the most intuitive
- Resulting ontology is compatible with RDFS and OWL Full, but not OWL DL or OWL Lite
- The subjects are in a hierarchy (AfricanLion isA Lion isA Animal)
  - Application can use this hierarchy to find books about Lion as well as books about its sub-subject; AricanLion
- Good approach if:
  - Want to keep things simple
  - Don't mind being in OWL Full
  - Don't mind using the class hierarchy as book subject

Approach 2:
Using special instances

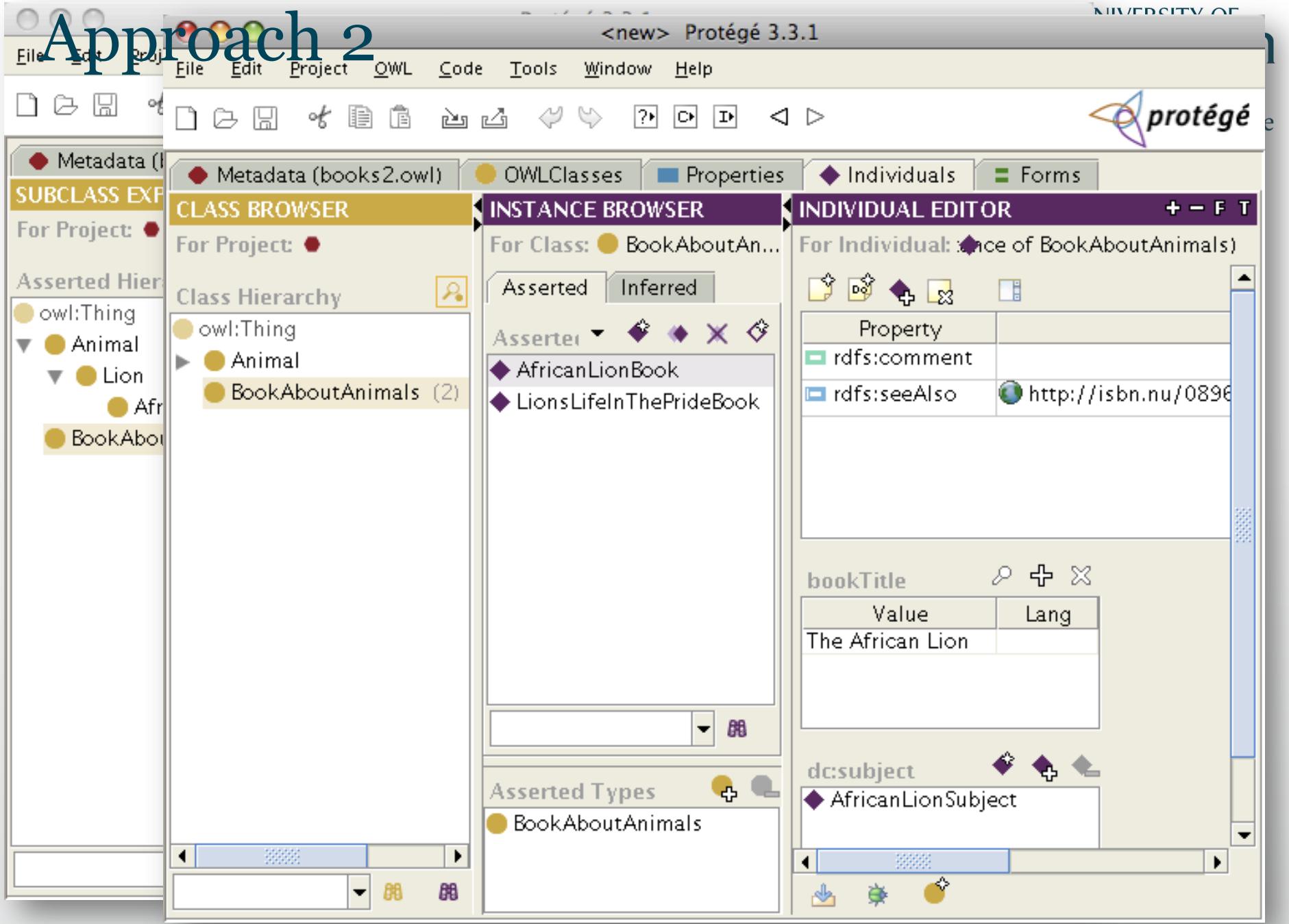
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Use instances of classes as property values



Source: W3C

Approach 2

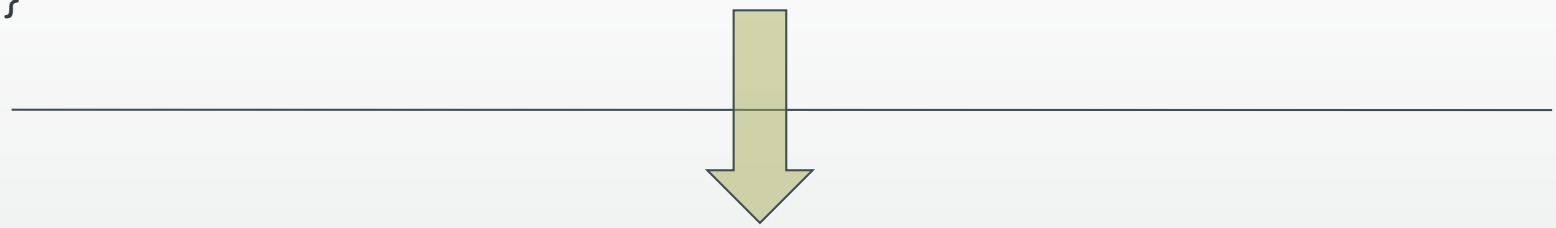


http://www.w3.org/TR/swbp-classes-as-values/books2.owl

UNIVERSITY OF Southampton
School of Electronics and Computer Science

SPARQL Query

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
PREFIX base:  <http://protege.stanford.edu/swbp/books2.owl#>

SELECT ?book
WHERE {
        ?book dc:subject ?subject .
        ?subject rdf:type base:Lion
}
```



```
<http://protege.stanford.edu/swbp/books2.owl#LionsLifeInThePrideBook>
<http://protege.stanford.edu/swbp/books2.owl#AfricanLionBook>
```

Notes on Approach 2

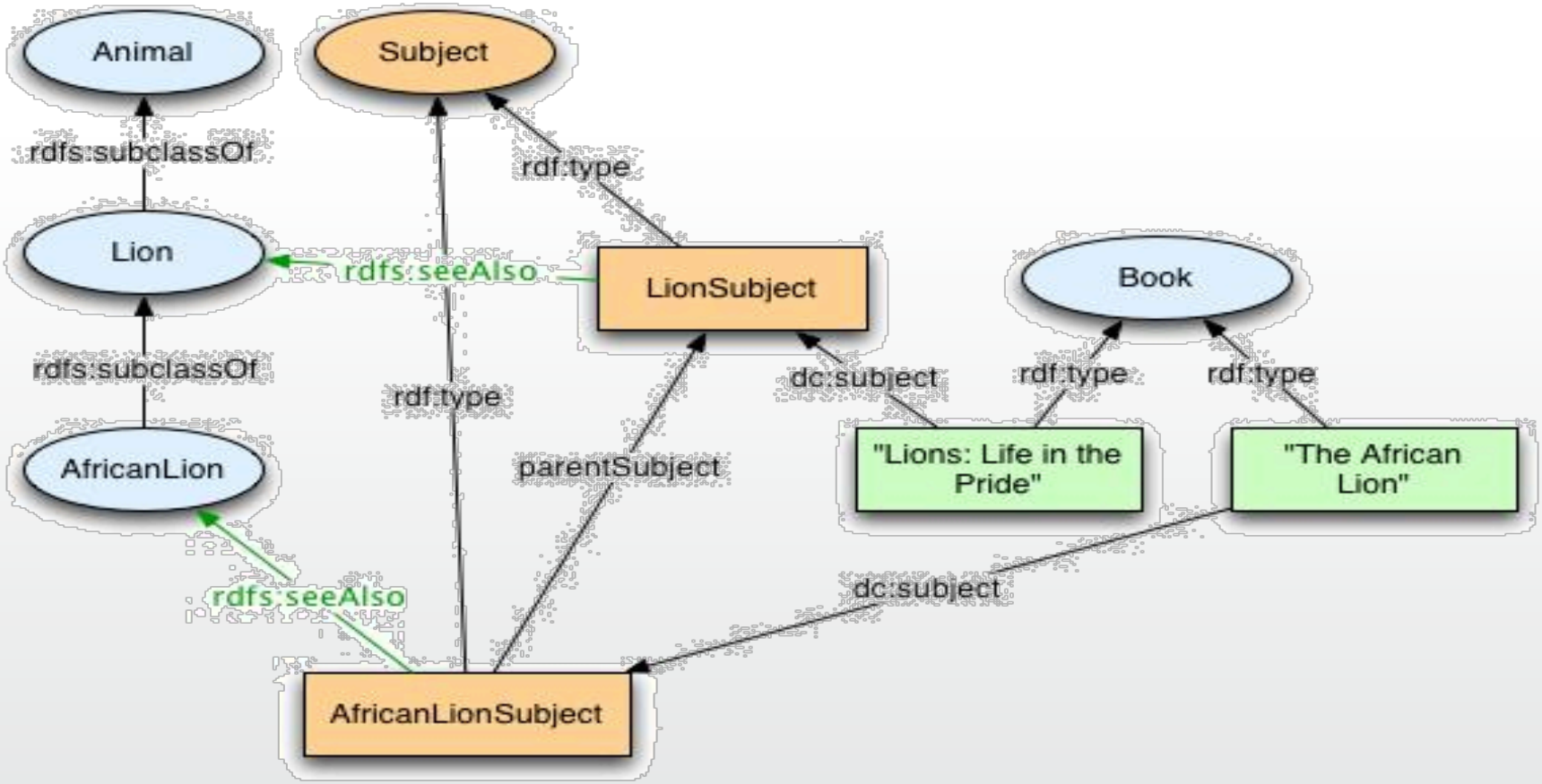
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Classes are not used as values directly
  - Using their instances as property values instead
- Ontology is compatible with OWL DL and OWL Lite
- We used the class Lion for the subject lion
  - Need a different one to refer to actual lions!
  - Shouldn't use the same concept for two conceptually different things
  - We need to be extra careful if the Animal ontology is important
    - Changing the meaning of classes may cause some interpretation problems
- No direct relation between the subjects
  - But the instance AfricanLionSubject is also an instance of Lion
- Use this approach if:
  - Want to stick to OWL DL or OWL Lite
  - Won't be changing the original meaning of any of the classes
  - Not concerned with the subjects not having direct links

Approach 3: Using a parallel instance hierarchy

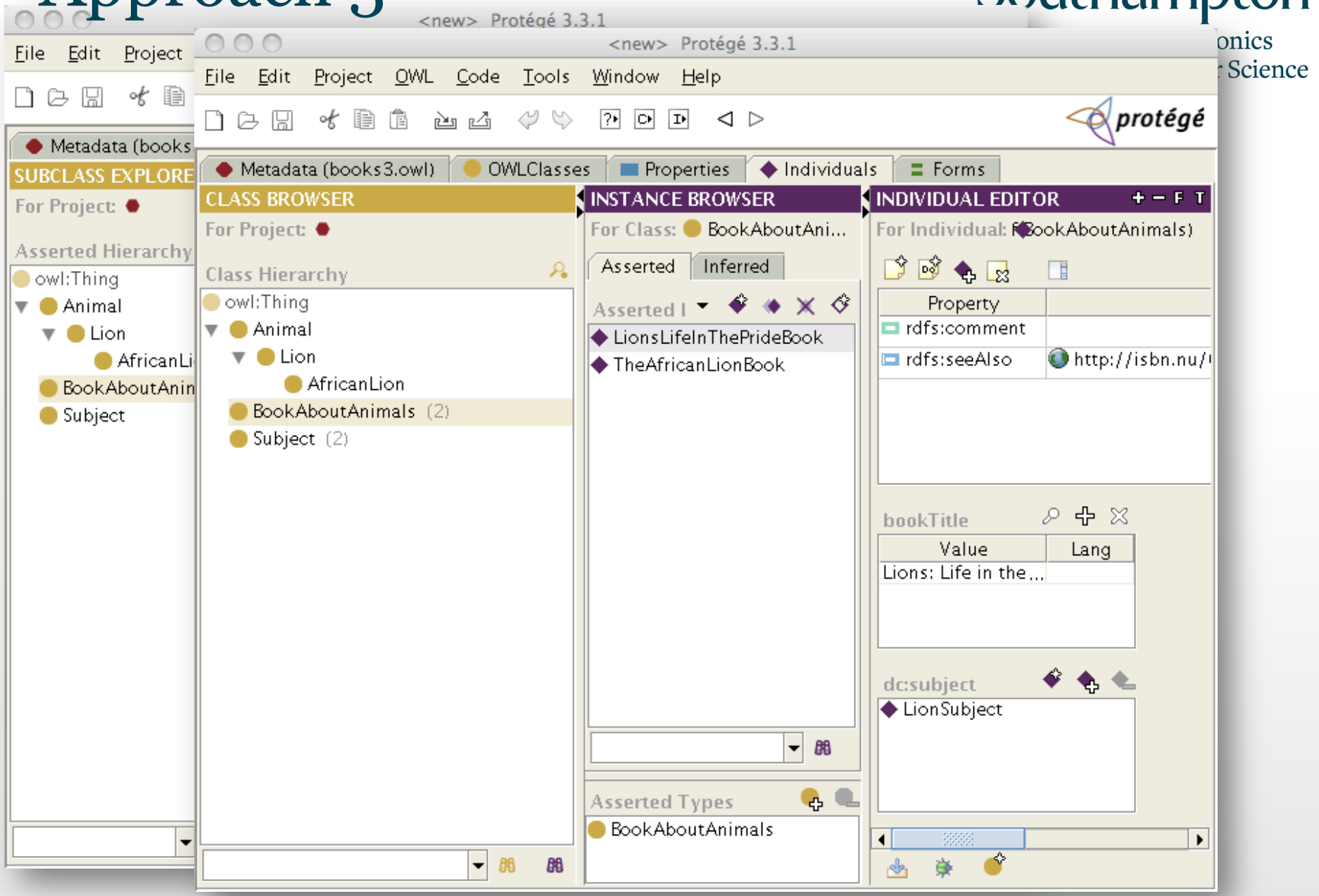
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Create a separate *subject* class



Source: W3C

Approach 3



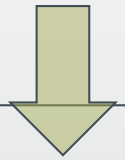
http://www.w3.org/TR/swbp-classes-as-values/books3.owl

SPARQL Query

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- One way of querying this model is by using the *seeAlso* annotation property.
- You can also query the transitive *parentSubject* property

```
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX base:  <http://protege.stanford.edu/swbp/books3.owl#>

SELECT ?book
WHERE {    ?book dc:subject ?subject .
          ?subject rdfs:seeAlso ?class .
           ?class rdfs:subClassOf base:Lion
     }
```



```
<http://protege.stanford.edu/swbp/books3.owl#LionsLifeInThePrideBook>
<http://protege.stanford.edu/swbp/books3.owl#AfricanLionBook>
```

Approach 3

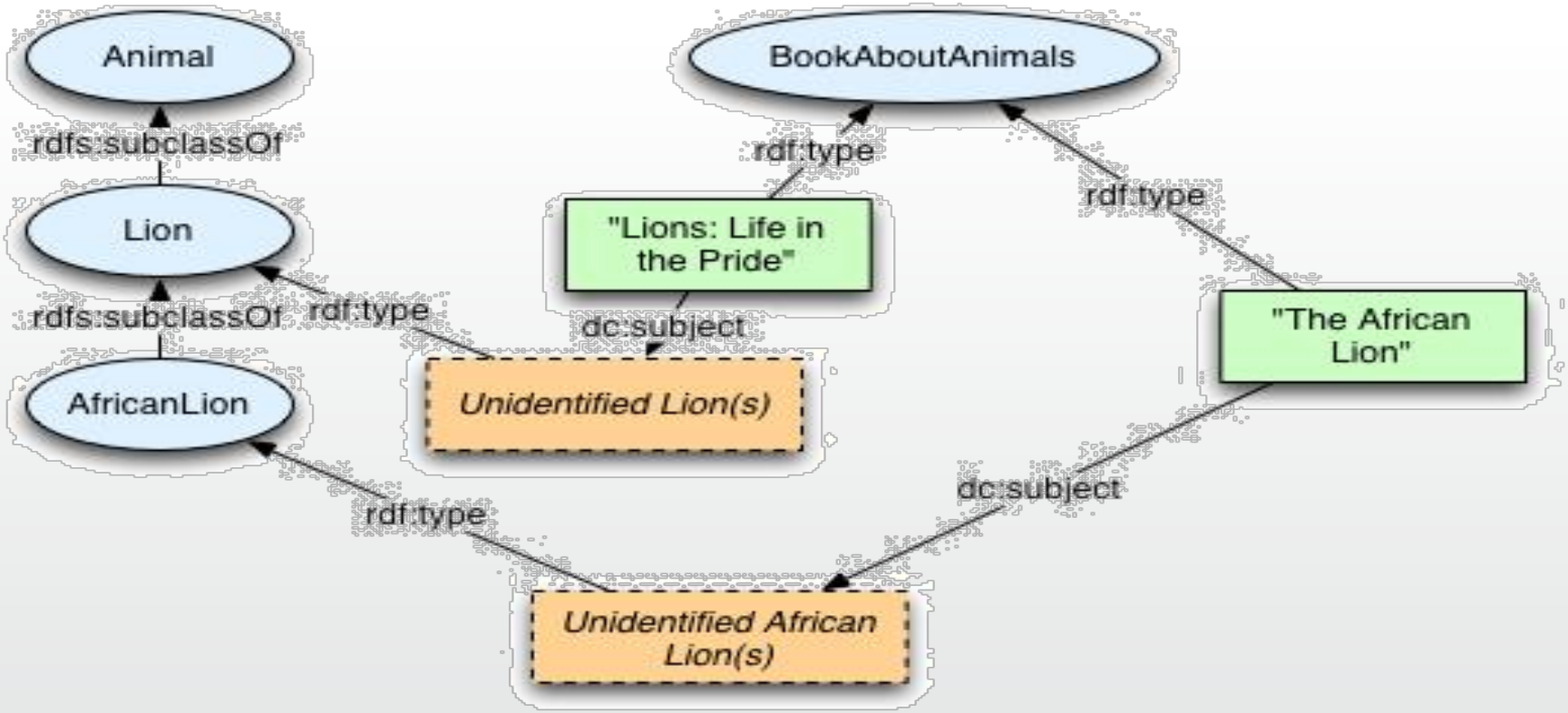
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Compatible with OWL DL and OWL Lite
  - Using classes as values for annotation properties (eg rdfs:seeAlso) does not change OWL DL compatibility
- The subject hierarchy can be recreated using the *parentSubject*
  - This property is transitive
  - Most reasoners can infer the *parentSubject* transitive property
    - But they won't be able to infer that a book about LionSubject is also about Animals
- Semantics for Lion and for the Lion subject are preserved
- The Animal and Subject hierarchies are independent of each other
- Maintenance is increased
  - Need to make sure all these classes and instances are consistent
- Use if:
  - Need to stay in OWL DL
  - Need to reason over the subject hierarchy
  - Not bothered by having parallel hierarchies

Approach 4
Using special restrictions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

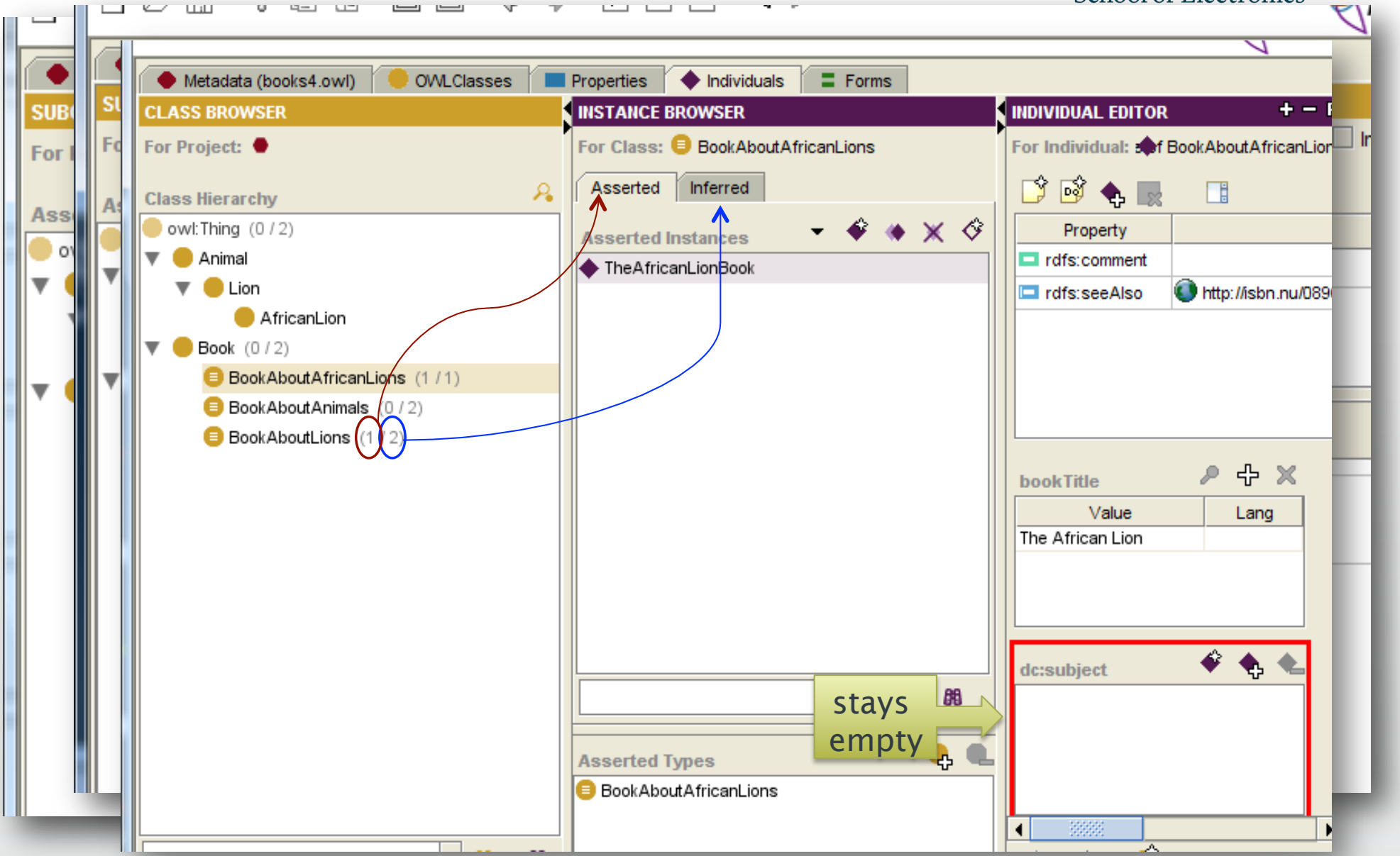
- Restrictions are used instead of specific values



Source: W3C

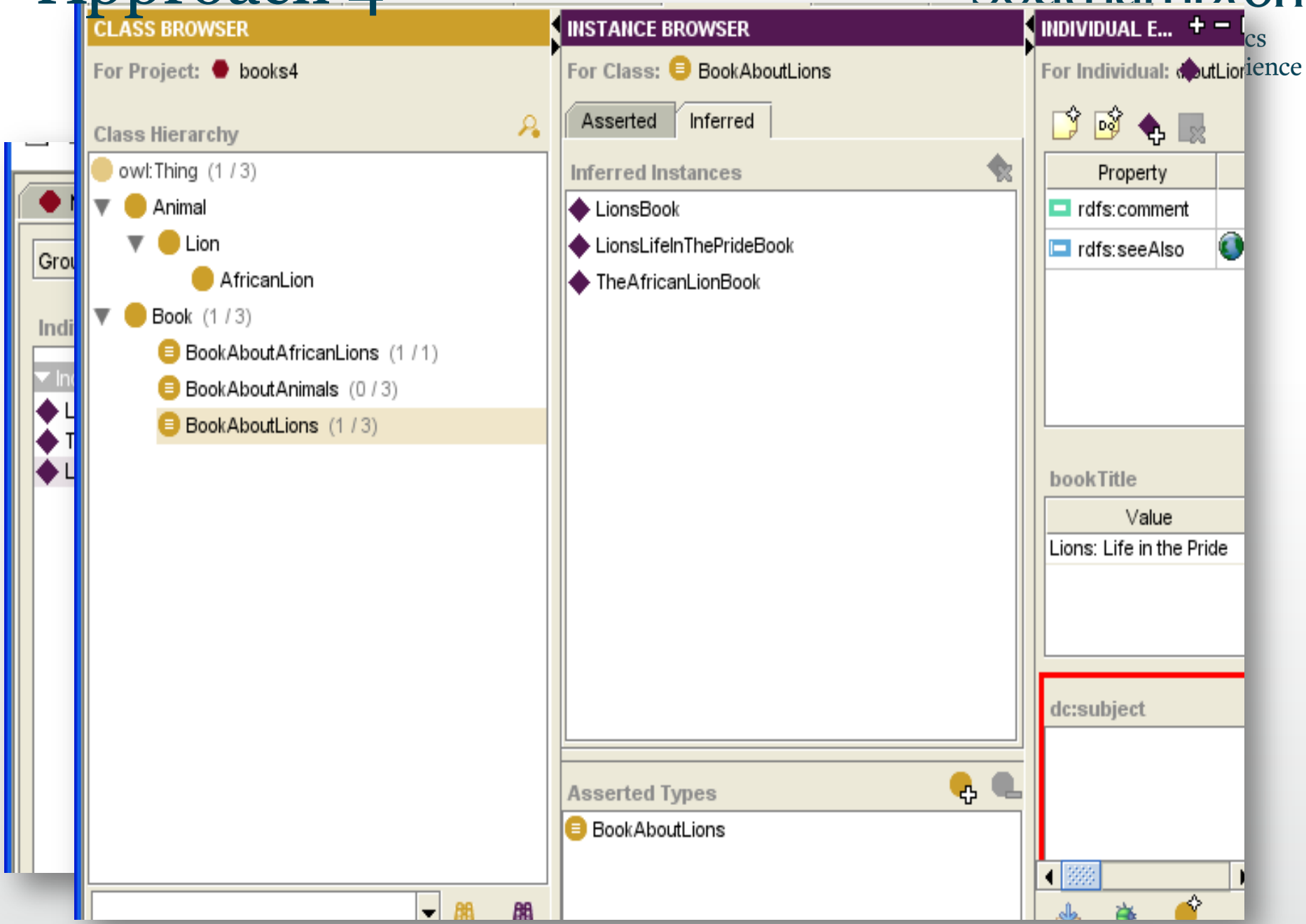
Approach 4

UNIVERSITY OF Southampton
School of Electronics



http://www.w3.org/TR/swbp-classes-as-values/books4.owl

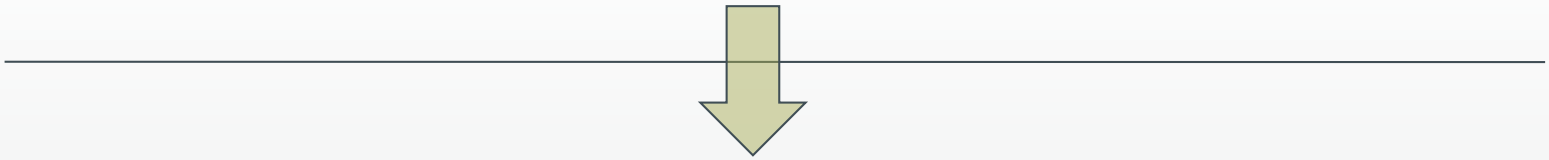
Approach 4


CLASS BROWSER
For Project: books4
Class Hierarchy
owl:Thing (1 / 3)
Animal
Lion
AfricanLion
Book (1 / 3)
BookAboutAfricanLions (1 / 1)
BookAboutAnimals (0 / 3)
BookAboutLions (1 / 3)
INSTANCE BROWSER
For Class: BookAboutLions
Asserted
Inferred
Inferred Instances
LionsBook
LionsLifeInThePrideBook
TheAfricanLionBook
Asserted Types
BookAboutLions
INDIVIDUAL E...
For Individual:
Property
rdfs:comment
rdfs:seeAlso
bookTitle
Value
Lions: Life in the Pride
dc:subject

SPARQL Query

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
PREFIX dc:  <http://purl.org/dc/elements/1.1/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX base:  <http://protege.stanford.edu/swbp/books3.owl#>


SELECT ?book

WHERE { ?book rdf:type base:BookAboutLions }
```



```
<http://protege.stanford.edu/swbp/books4.owl#LionsLifeInThePrideBook>
<http://protege.stanford.edu/swbp/books4.owl#AfricanLionBook>
```

- Only the first book will be returned if no reasoner is used

Approach 4

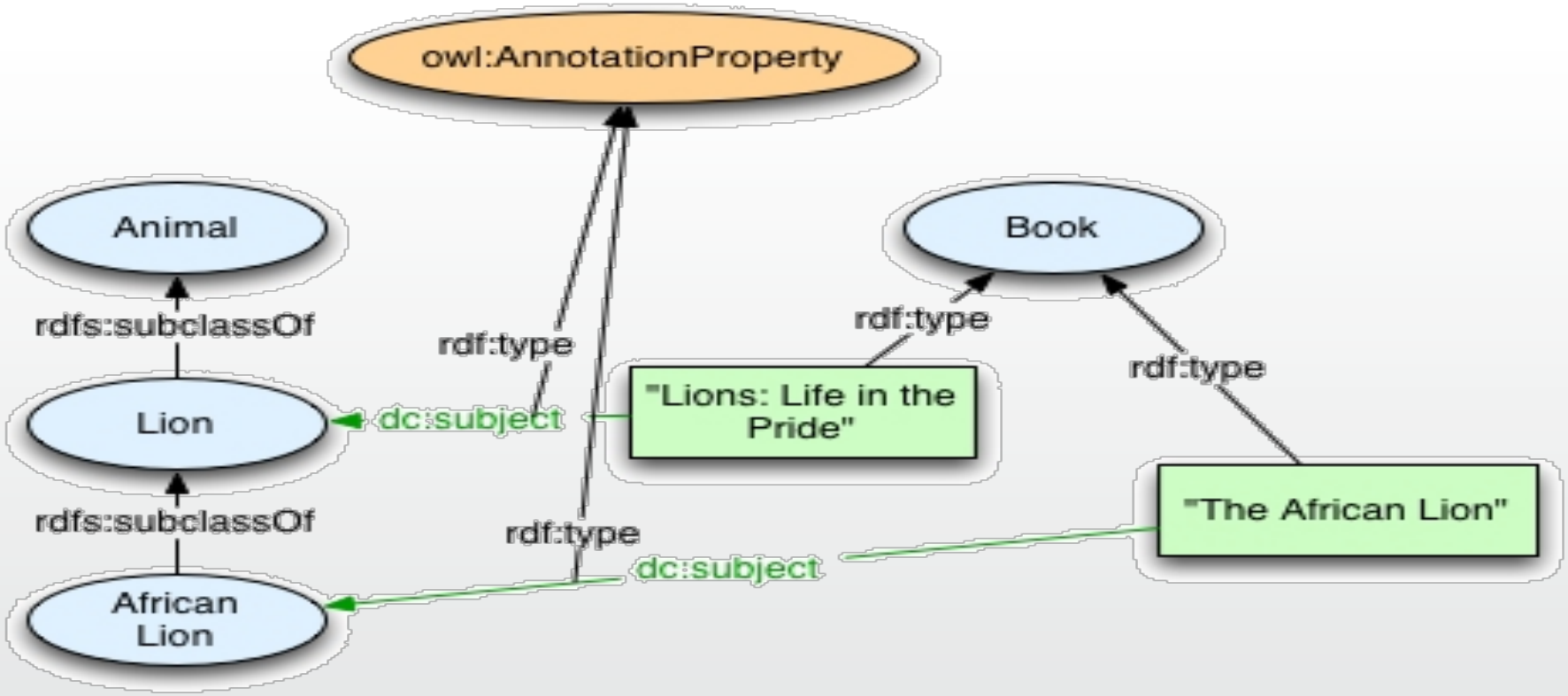
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Compatible with OWL DL
- A reasoner can infer that a book with *subject* Lion also has the subject Animal
  - Can use a DL reasoner to classify specific books
- Subjects are assigned to books by creating instances of the relevant book subject class
  - No need to explicitly set any subject values
  - Can also use unspecified individuals of the class as property values, rather than the class itself
  - Interpretation: the subject is a prototypical lion, rather than the Lion class
- Use if:
  - Want to be in OWL DL
  - Want to use DL reaonsers to classify your ontology

Approach 5: Using annotation properties

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Link individuals of Book with subjects using an annotation property



Source: W3C

Approach 5

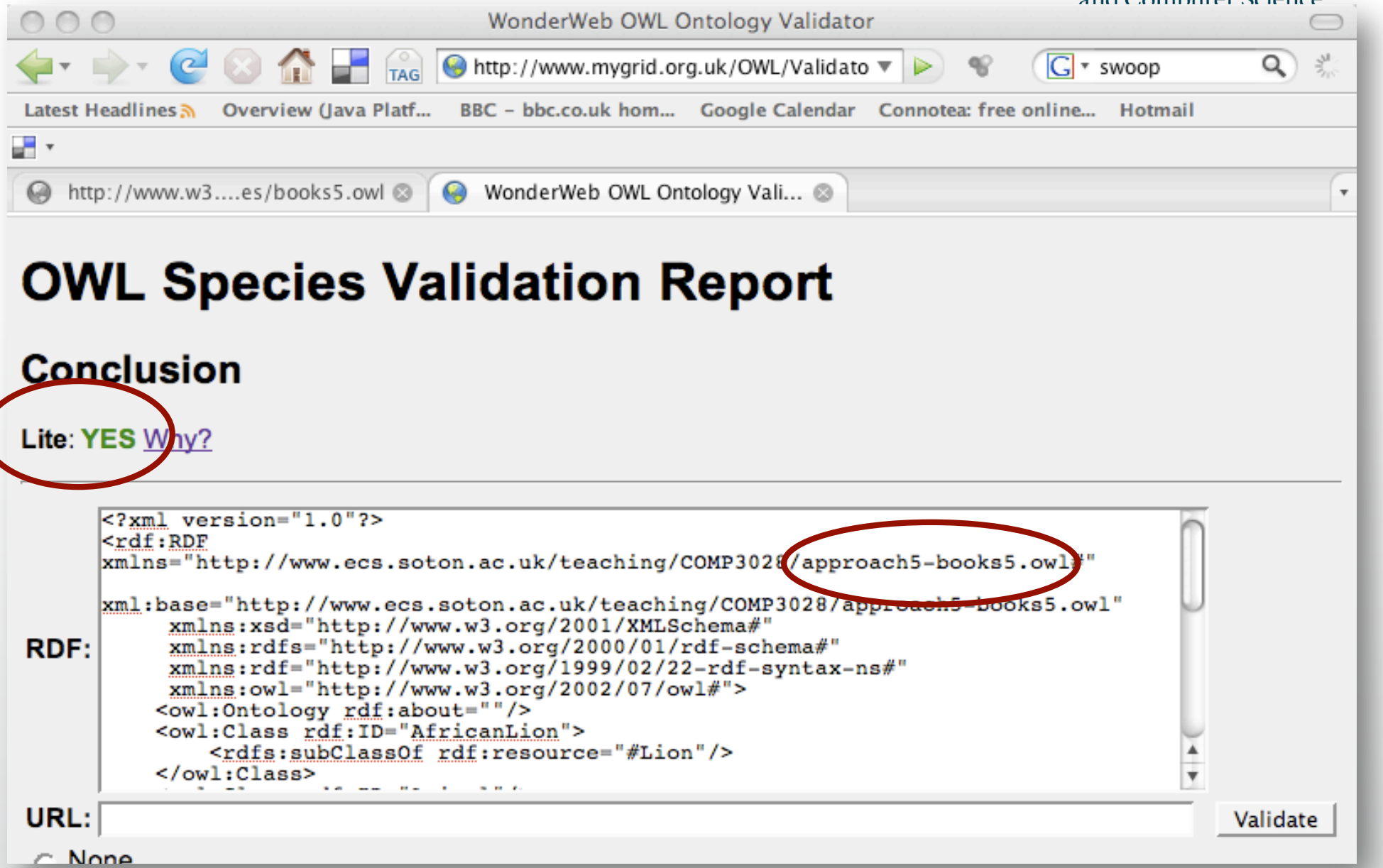
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Implementing this ontology in Protégé turns the ontology into OWL:FULL
  - Because the property becomes both owl:ObjectProperty and owl:AnnotationProperty
- Better to write/fix it by hand
- Download it from:

```
http://users.ecs.soton.ac.uk/ha/teaching/COMP3028/
   approach5-books5.owl
```

Validating the Ontology

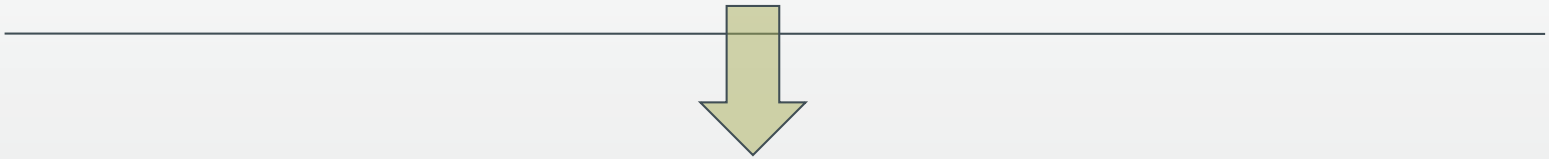
UNIVERSITY OF Southampton
School of Electronics and Computer Science



SPARQL Query

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX base:  <http://www.ecs.soton.ac.uk/teaching/COMP3028/approach5-
   books5.owl#>


SELECT ?book

WHERE { ?book rdf:type base:Book .

        ?book base:subject ?class .

        ?class rdfs:subClassOf base:Lion }
```



```
<http://www.ecs.soton.ac.uk/teaching/COMP3028/approach5-
books5.owl#LionsLifeInThePrideBook>
<http://www.ecs.soton.ac.uk/teaching/COMP3028/approach5-
books5.owl#AfricanLionBook>
```

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Approach 5

- Compatible with OWL DL
  - Annotation properties can have classes as values in OWL DL
- Annotation properties cannot have different types
  - *dc:subject* cannot be an annotation property and an object or datatype property
  - This will render the ontology OWL FULL
- Restrictions cannot be applied to annotation properties
- DL reasoners don't use annotation values

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Topics

- N-ary relations
- Classes as property values
- Value partitions and Value sets

UNIVERSITY OF Southampton
School of Electronics and Computer Science

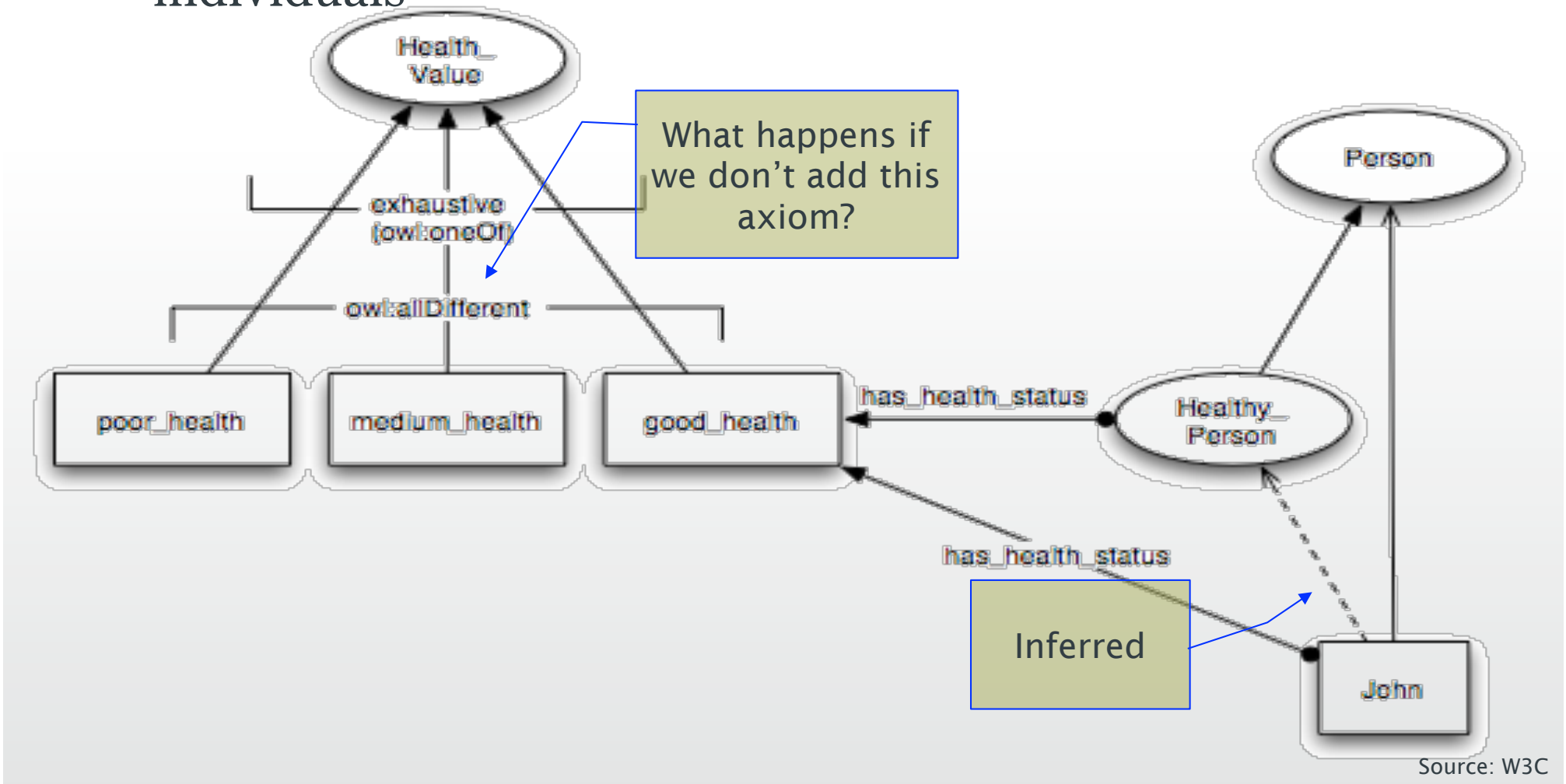
Value Partition

- Descriptive features are quite common in ontologies
- Examples:
  - Size {small, medium, large}
  - Risk {dangerous, risky, safe}
  - Health status {good health, medium health, poor health}
- Also called “qualities”, “modifiers”, “attributes”
- A property can have only one value for each feature to ensure consistency
- Such features can be represented as:
  - Enumerated individuals
  - Disjoint classes
  - Datatype values

Approach 1
Values as sets of individuals

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Class Health_Value is an enumeration of three individuals



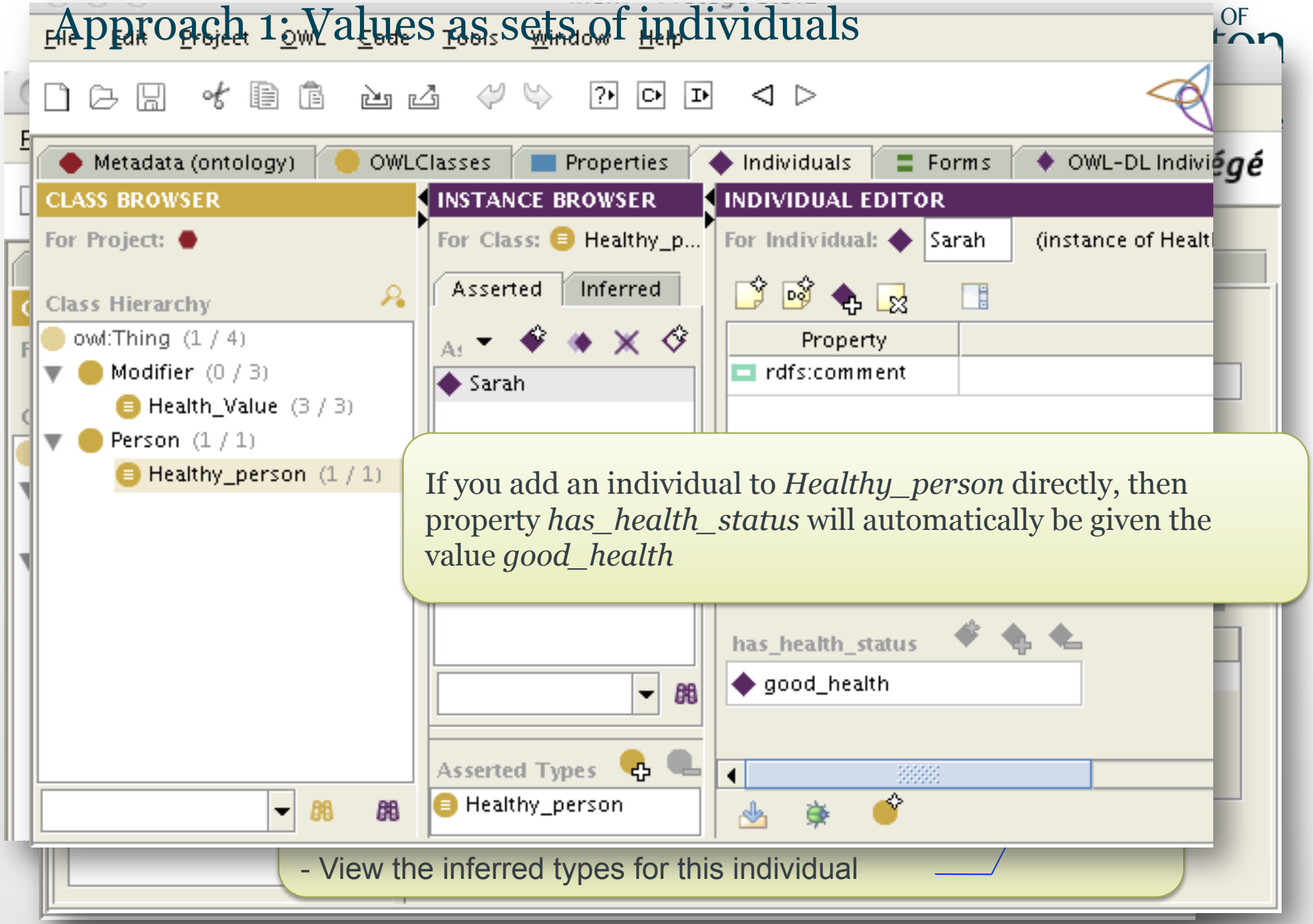
Source: W3C

For Geeks Only

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
:has_health_status
      a       owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:range :Health_Value .
John
      a       :Person ;
      :has_health_status :good_health .

:good_health
      a       :Health_Value .

:Healthy_person
      a       owl:Class ;
      owl:equivalentClass
              [ a       owl:Class ;
                owl:intersectionOf (:Person [ a       owl:Restriction ;
                            owl:hasValue :good_health ;
                            owl:onProperty :has_health_status
                          ])
              ] .
```

Approach 1: Values as sets of individuals



http://www.w3.org/TR/swbp-specified-values/values-as-individuals-01.owl

Approach 1: Values as sets of individuals

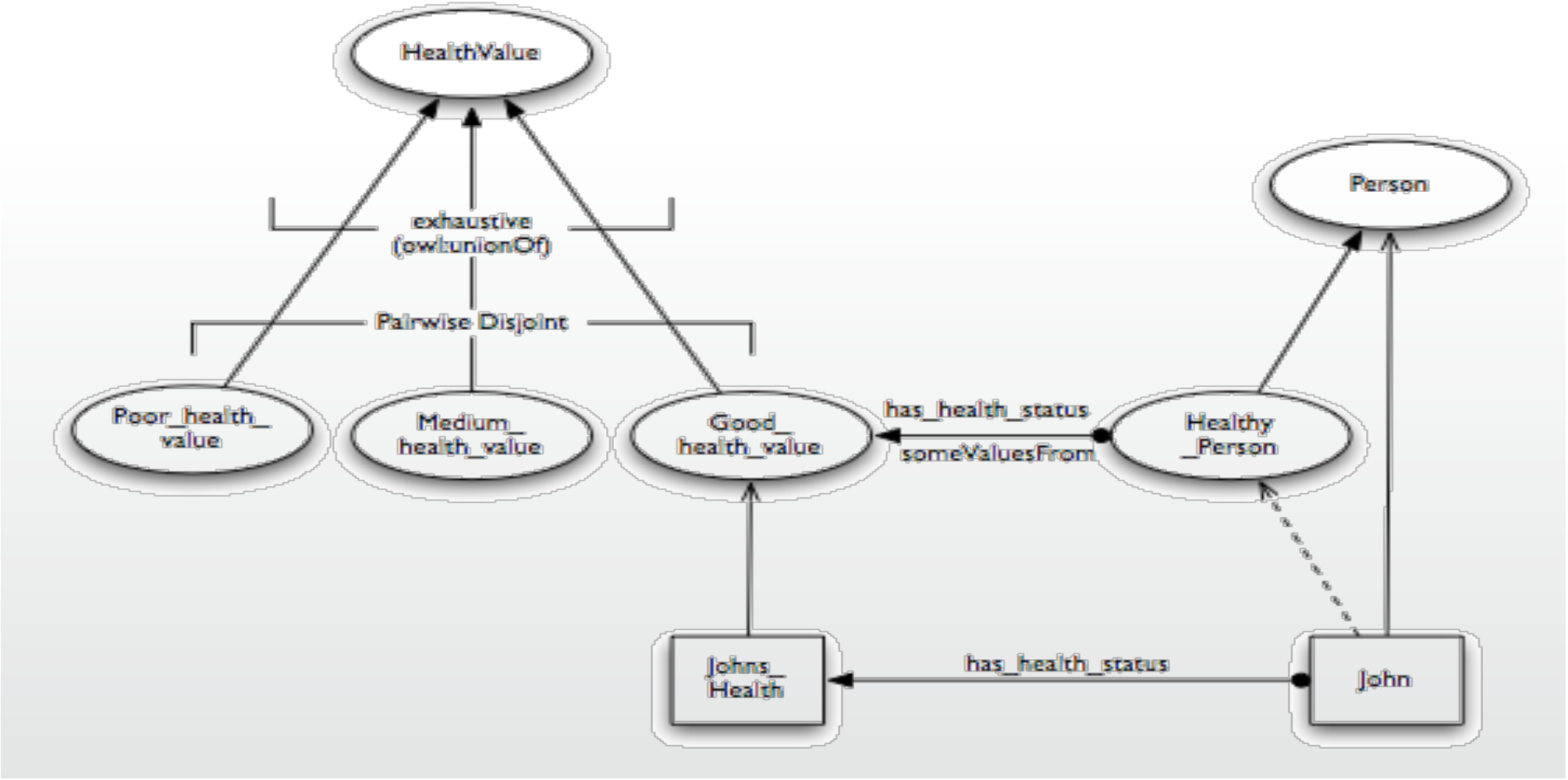
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Need an axiom to set the three health values to be different from each other
  - This way, a person cannot have more than one health value at a time
- Values cannot be further partitioned
  - Eg we cannot have moderately_good_health as a subtype of good_health
  - Only equality and difference between individuals is allowed in OWL
- Only one set of values is allows for a feature
  - The class cannot be equivalent to more than one set of distinct values
  - Doing so will cause inconsistencies
- OWL DL compatible

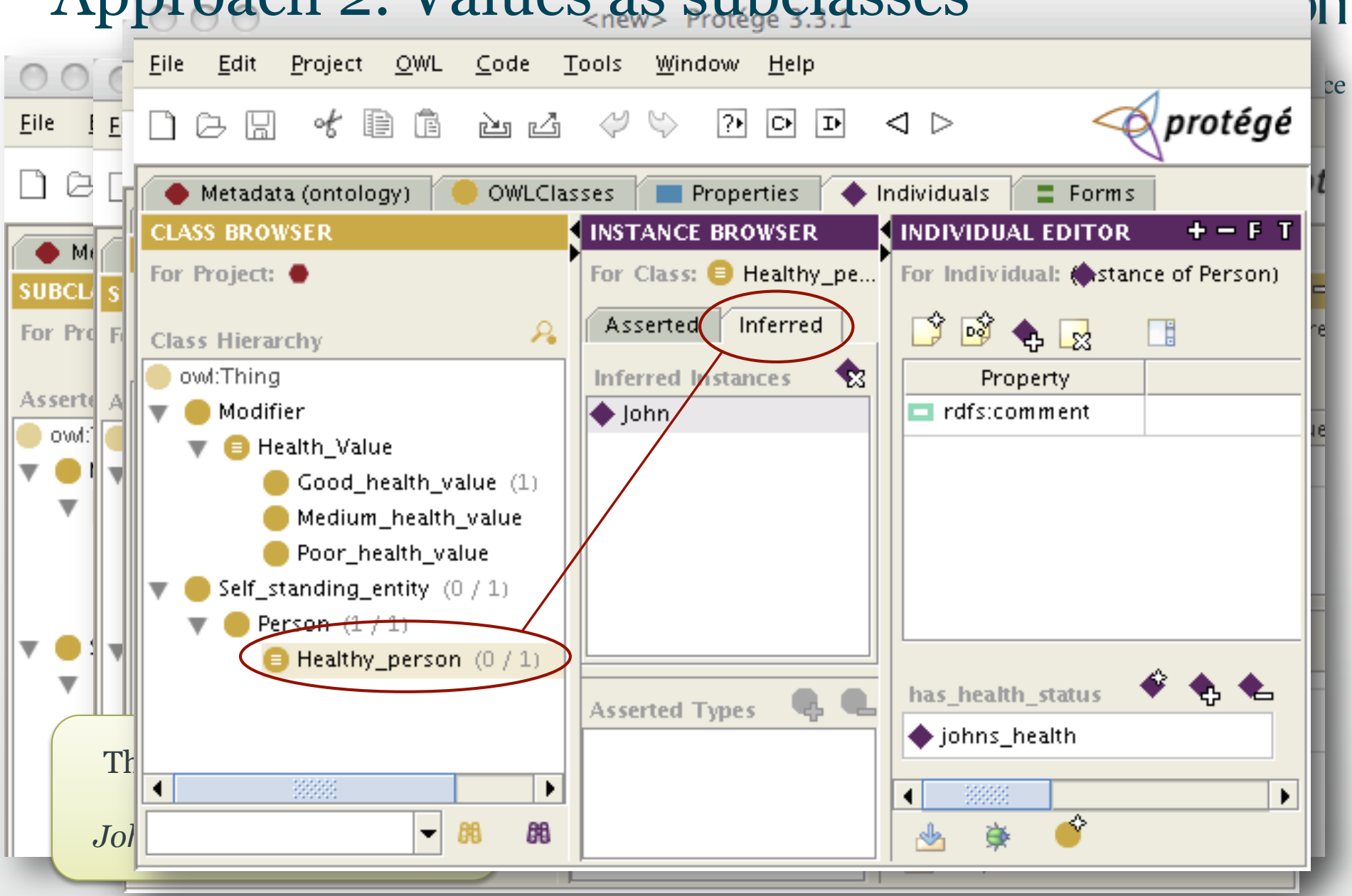
Approach 2: Values as subclasses

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Values are disjoint subclasses



Approach 2: Values as subclasses

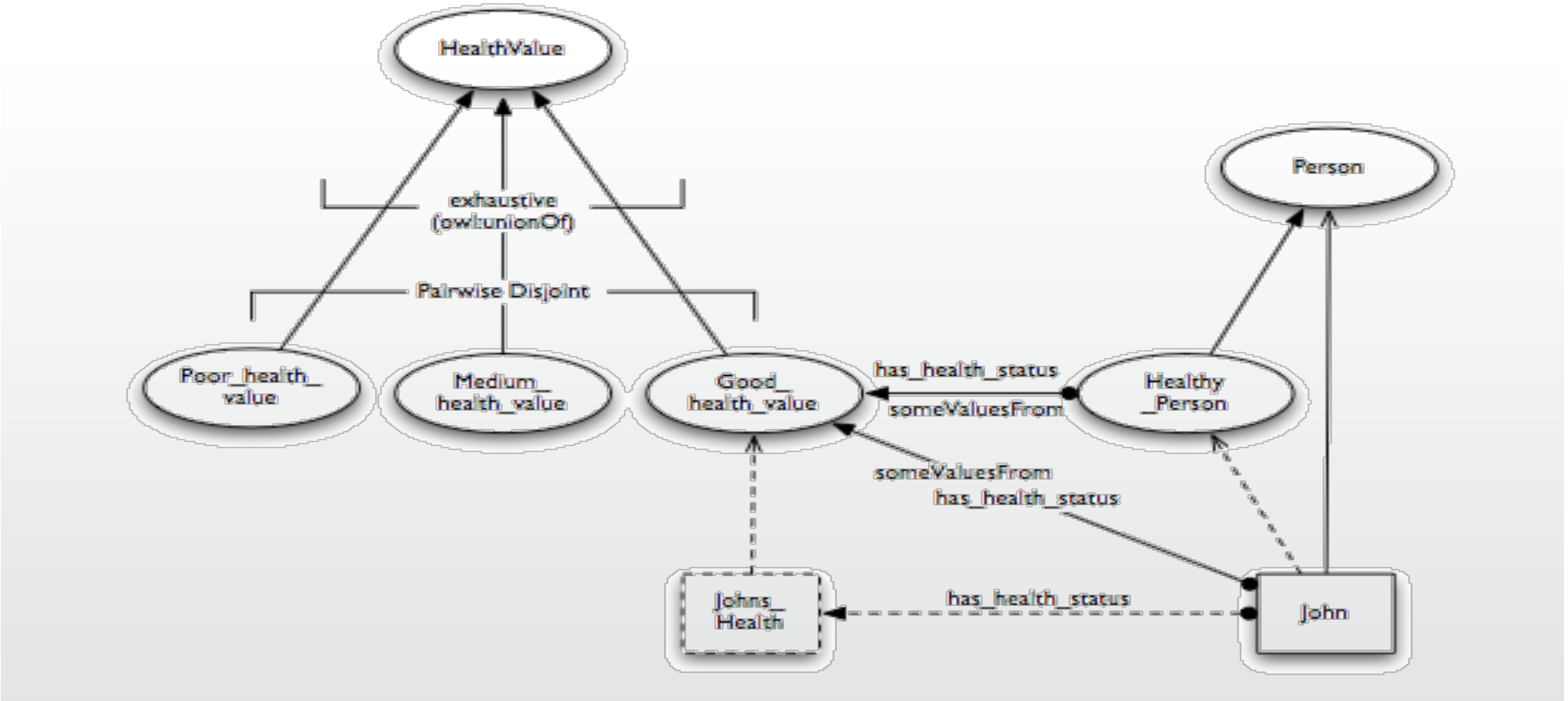


http://www.w3.org/TR/swbp-specified-values/value-partitions-variant-1.owl

Approach 2: Values as subclasses

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- The instance Johns_Health can be made anonymous



UNIVERSITY OF Southampton
School of Electronics and Computer Science

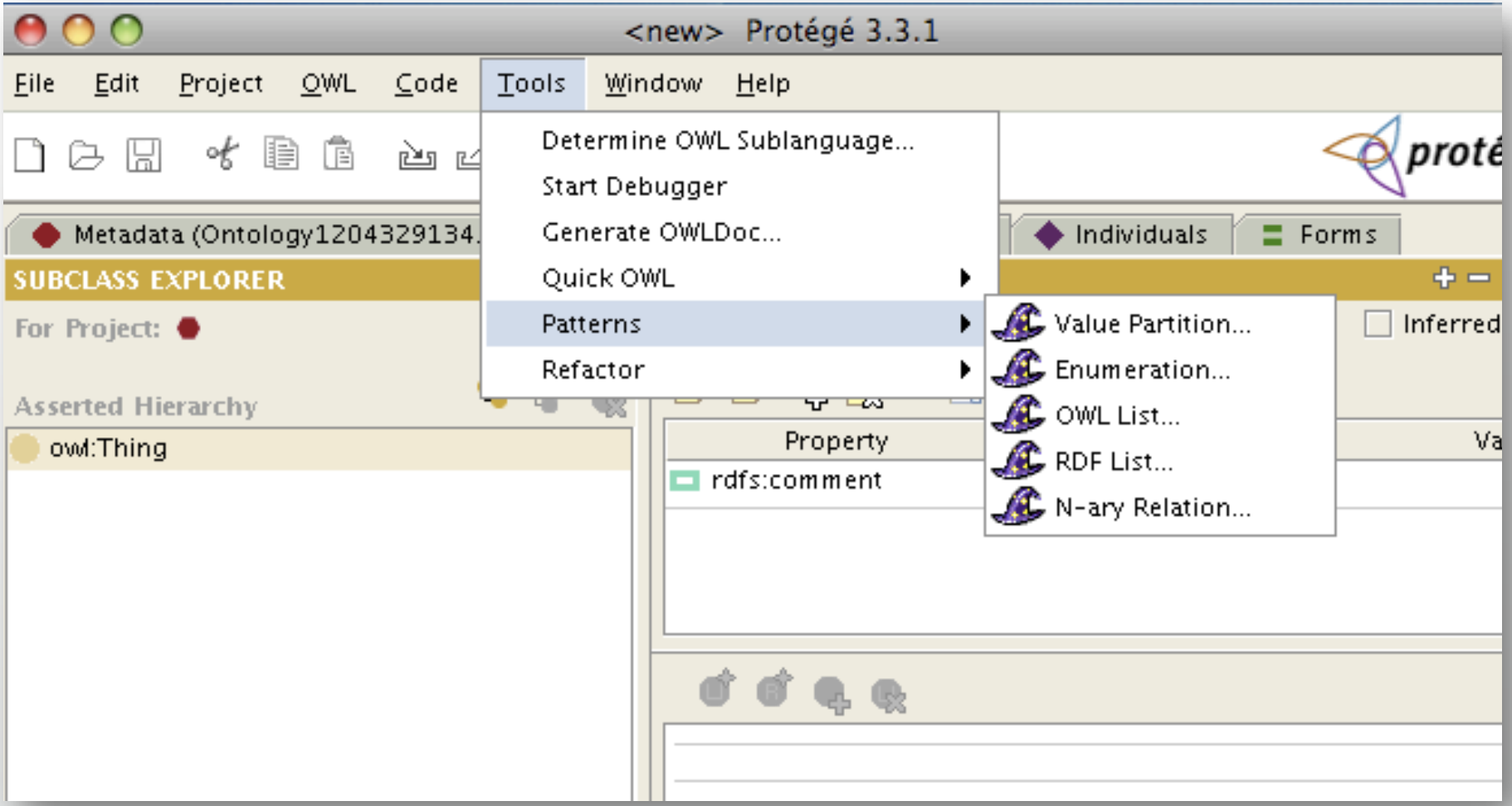
Approach 2: Values as subclasses

- OWL DL compatible
- DL reasoners can classify the ontology
- Values can be further partitioned
  - Simply add subclasses to the value classes
- Can have alternative partitioning of the same feature

OWL Wizards

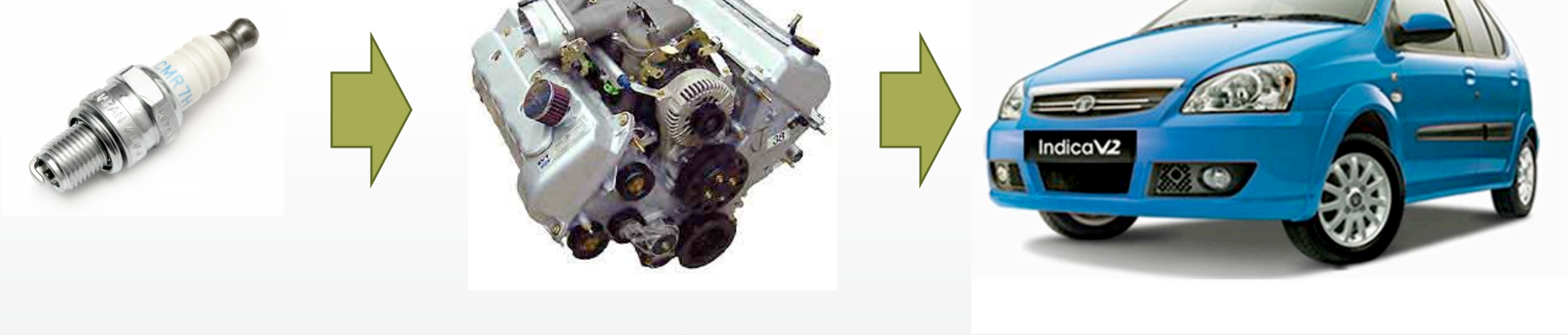
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Protégé has OWL wizards for creating n-ary relations, value partitions and enumerations (values as individuals)



Meronymies (part-whole relations)

- Taxonomies are not the only hierarchical relation that we wish to model



- A spark plug isn't a kind of engine (class-instance)
- A spark plug is a **part of** an engine

Simple Part-Whole Representation

- We need two properties:
  - partOf (a transitive property)
  - directPartOf (a subproperty of partOf)

Part-Whole Hierarchies

- Represent part-whole relationships between classes using someValuesFrom restrictions

$$\text{SparkPlug} \sqsubseteq \exists\ \text{directPartOf.Engine}$$

$$\text{Engine} \sqsubseteq \exists\ \text{directPartOf.Car}$$

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Defining Classes of Parts

- Extend the ontology with classes of parts for each level
  - Reasoner can automatically derive a class hierarchy

$$\text{CarPart} \equiv \exists\ \text{partOf.Car}$$

$$\text{DirectCarPart} \equiv \exists\ \text{directPartOf.Car}$$

$$\text{EnginePart} \equiv \exists\ \text{partOf.Engine}$$

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Fault location

- Allows reasoner to conclude that a fault in a part is a fault in a whole
- Need a new property for the location of a fault: hasLocus
- Need a new class for faults: Fault

$$\text{FaultInCar} \equiv \text{Fault} \sqcap \exists\ \text{hasLocus.CarPart}$$

$$\text{FaultInEngine} \equiv \text{Fault} \sqcap \exists\ \text{hasLocus.EnginePart}$$