Lucia Pifferi
Matricola 537849

Semantic web exam

Q9

Identify two different assertions that would make the ontology inconsistent.

1. book1 hasPublisher publisher1
   book1 hasPublisher publisher2
   Functional: hasPublisher
   publisher1 DifferentFrom publisher2

2. HumanCharacter DisjointWith NotHumanCharacter
   participant1 Type HumanCharacter
   participant1 Type NotHumanCharacter

Q10

Define the complex role inclusion axiom capturing the fact that if a narrator creates a narrative that is reported in a book that is published by a publisher, then the narrator collaborates with that publisher.

I've defined the complex role inclusion axiom inside the ontology.

V_C = {:Narrator, :Narrative, :Book, :Publisher}
V_{OP} = {:creates, :isReportedIn, :hasPublisher, :collaborates}

SubObjectPropertyOf(ObjectPropertyChain(:creates :isReportedIn :hasPublisher) :collaborates)

(creates **o** isReportedIn **o** hasPublisher) $\subseteq$ collaborates

(narrator1, narrative1) **o** (narrative1, book1) **o** (book1, publisher1) → (narrator1, publisher1)

Q11

Verify if the created ontology (including the complex role inclusion axiom defined in Q10) satisfies the global restrictions on the axioms of an OWL 2 DL ontology.

- The restriction on owl:topDataProperty is satisfied because the ontology does not include any axiom on owl:topDataProperty.
- The restrictions on Datatypes are satisfied because:
  - the ontology only uses datatypes from the OWL 2 datatype map;
  - no data ranges have been defined.
- The restriction on Simple Roles is satisfied because no composite object property is used in an axiom of the forbidden kinds.
- The restriction on Property Hierarchy is satisfied because there is only one property chain in the ontology, so it is impossible to generate cycles.
- The restrictions on Anonymous Individuals are satisfied because there are no anonymous individuals in the ontology.

Q12

Write the following queries in SPARQL:

12.1 *Find how many events occurred in fictional locations, grouped by location.*

```
prefix : <http://www.semanticweb.org/luciapifferi/ontologies/2020/semantic_web_exam/narratives#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT (COUNT(?event) AS ?nEvents) ?location
WHERE{?event rdf:type :Event;
             :occursIn ?location.
      ?location rdf:type :FictionalLocation.
     }
GROUP BY ?location
```

12.2 *Find all events that totally occurred between 1 January 2018 (included) and 31 December 2019 (included).*

```
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix : <http://www.semanticweb.org/luciapifferi/ontologies/2020/semantic_web_exam/narratives#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?event
WHERE{?event rdf:type :Event;
             :hasBeginning ?beginDate;
             :hasEnding ?endDate.

     FILTER(?beginDate >= "2018-01-01T00:00:00"^^xsd:dateTime && ?endDate < "2020-01-01T00:00:00"^^xsd:dateTime)
     }
```

12.3 *Find all the events that do not have any human participants.*

```
prefix : <http://www.semanticweb.org/luciapifferi/ontologies/2020/semantic_web_exam/narratives#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?event
WHERE{?event rdf:type :Event.
        MINUS{
            ?event :hasParticipant ?participant.
             ?participant rdf:type :Human. }
     }
```

12.4 *Find all the narratives that are published in a book, along with the title of the book, the ISBN code of the book and the publisher of the book.*

```
prefix : <http://www.semanticweb.org/luciapifferi/ontologies/2020/semantic_web_exam/narratives#>

SELECT ?narrative ?bookTitle ?ISBNcode ?publisher
WHERE{
  ?book :reports ?narrative;
        :hasPublisher ?publisher;
        :hasTitle ?bookTitle;
        :hasISBNcode ?ISBNcode.
}
```

12.5 *Find all the distinct events that have a non-human participant or occur in a fictional location.*

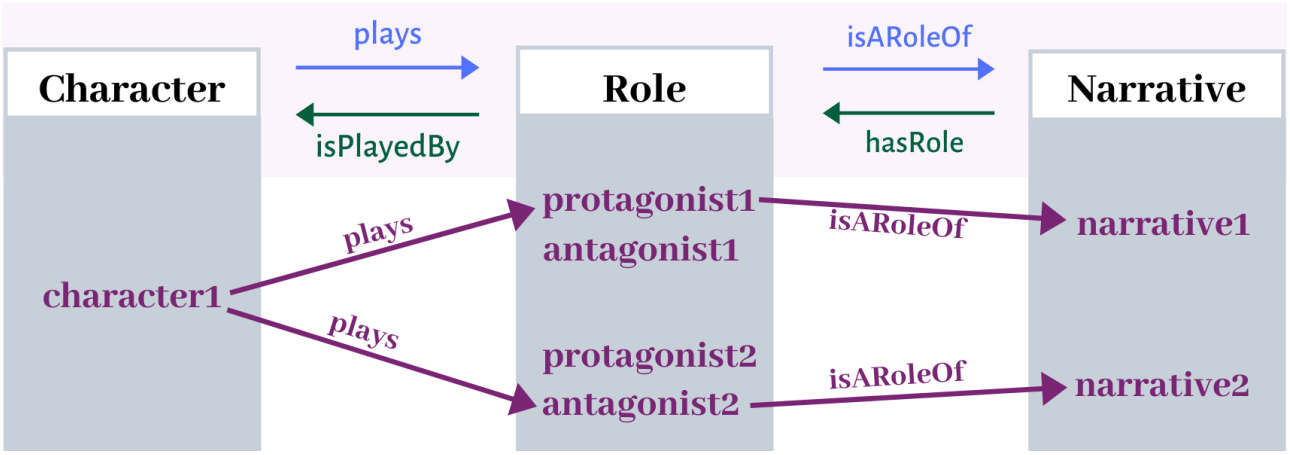
```
prefix : <http://www.semanticweb.org/luciapifferi/ontologies/2020/semantic_web_exam/narratives#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT ?event
WHERE{
  ?event rdf:type :Event;
         :hasParticipant ?participant;
         :occursIn ?location.

  {?participant rdf:type :NotHuman .}
  UNION
  {?location rdf:type :FictionalLocation .}
}
```

Q13

Suppose that you want to model the different roles of a character in different narratives (e.g. the same character is a protagonist in narrative 1 and an antagonist in narrative 2). How would you model this situation?

I would create three classes (*Character*, *Role*, *Narrative*) and two properties (*plays* and *isARoleOf*) with their respective inverses (*isPlayedBy* and *hasRole*).
The property *plays* links the character to the role, such that a character plays minimum one role (I would add an axiom to state that). *IsPlayedBy* is the inverse property of *plays* and it's functional because – in my conceptualization - a role can be played only by one character (and so *plays* is inverse functional).
The property *isARoleOf* links the role to the narrative and it's functional because a role can be the role of only one narrative. The inverse property of *isARoleOf* is *hasRole* and it is inverse functional. *hasRole* has Narrative as domain and Role as range. A narrative would have minimum one role, so I would add the axiom: Narrative *hasRole* min 1 Role. To supplement the information supplied so far, I would also add the axiom: Role *isARoleOf* exactly 1 Narrative.

The graph below better shows every relationship. I reported object properties in blue and their inverses in green. Furthermore, I populate every class with individuals (in purple). protagonist1, antagonist1, protagonist2 and antagonist2 are different individuals. And so are narrative1 and narrative2.



I could also create a complex role inclusion axiom that captures the fact that if a character plays a role and that role is a role of a narrative, then the character *is a character of* that narrative.

```
:isACharacterOf rdf:type owl:ObjectProperty ;
                rdfs:domain :Character ;
                rdfs:range :Narrative ;
                owl:propertyChainAxiom ( :plays
                                         :isARoleOf
                                       ) ;
                rdfs:label "isACharacterOf"@en .
```