Esercizio 8

Parte 1:

Definire una mini-ontologia relativa al dominio universitario nel quale siano *almeno* rappresentati i concetti di corso, studente, professore, esame (di uno studente, relativo ad un corso, avvenuto in una certa data e con un determinato esito). Con l'unico vincolo che ogni corso è insegnato da un solo professore, utilizzare e rappresentare le proprietà necessarie per relazionare i vari concetti tra loro.

Premessa alla soluzione:

Osservate come una appropriata rappresentazione del dominio (eventualmente guidata da una analisi delle informazioni che da esso andranno estratte e delle quali abbiamo un esempio nelle domande successive) può fornire una soluzione di facile comprensione senza bisogno di una complessa formulazione.

Soluzione parte 1:

L'ontologia può essere molto semplice

Elenco classi (nessuna relazione IS-A tra di esse):

Elenco proprietà:

corso_esame	dom: Esame	rng: Corso	
docente ↔ insegna	dom: Corso	rng: Professore	functional
esami_dati ↔ studente_esaminato	dom: Studente	rng: Esame	
insegna ↔ docente	dom: Professore	rng: Corso	
studente_esaminato ↔ esami_dat	dom: Esame	rng: Studente	functional
votazione	dom: Esame	rng: xsd:integer	functional
cognome	dom: Persona	rng: xsd:string	
data	dom: Esame	rng: xsd:date	functional
nome	dom: Persona	rng: xsd:string	

Commento alla soluzione

Come è possibile osservare, il concetto di Esame (di per se una relazione sul dominio) non è stato rappresentato attraverso una proprietà, bensì reificato in una classe, le cui tre proprietà: corso_esame, studente_esaminato, votazione, agiscono da ruoli della relazione, stabilendo una quaterna di valori:

Corso, **Studente**, *votazione*, *data*

Relativi alla prestazione (intero associato alla proprietà votazione) di uno studente (istanza di Studente) nell'esame di un determinato corso (istanza di Corso) svolto in una certa data.

Parte 2 (Concetti complessi, definizione in DL):
Rappresentare nell'ontologia, in termini di condizioni necessarie e sufficienti, la classe degli studenti che ha sostenuto l'esame di un determinato corso. (e.g. StudentiDiSBC)

Domanda Bonus: Rappresentare, in termini di condizioni necessarie e sufficienti, la classe degli studenti che ha sostenuto almeno un esame con un determinato professore (e.g. StudentiDiPazienza).

Soluzione parte 2:
∃ esami_dati (corso_esame ∋ Sistemi_basati_su_conoscenza)

Soluzione parte 2 bonus:
∃ esami_dati (∃ corso_esame (docente ∋ Pazienza))

Parte 3 (Interrogazione SPARQL):

a) Scrivere una query SPARQL che faccia una stampa di tutti gli esami (con voto) dati dallo studente con nome e cognome Mario Rossi. Attenzione, mentre nella ontologia c'è una strutturazione opportuna che separa le istanze di esame, il corso di riferimento etc… nella tabella di questo risultato, si intende avere ciò che nel parlar comune identifica un esame, ovvero il suo corso di riferimento

es:

Course	Vote
SBC	30
AI	27

Soluzione Parte 3 a)

```
PREFIX un: <http://art.uniroma2.it/ontologies/universita#>
SELECT ?course ?vote
FROM <http://art.uniroma2.it/ontologies/universita>
WHERE { ?stud un:nome "Mario" .
        ?stud un:cognome "Rossi" .
        ?stud un:esami_dati ?exam .
        ?exam un:corso_esame ?course .
        ?exam un:votazione ?vote .
 }
```

b) Scrivere una query SPARQL che faccia una stampa tipo libretto (data, esame, voto) di tutti gli esami dati dallo studente Mario Rossi e impartiti dalla prof. Maria Teresa Pazienza:

Soluzione Parte 3 b)

```
PREFIX un: <http://art.uniroma2.it/ontologies/universita#>
SELECT ?date ?course ?vote
FROM <http://art.uniroma2.it/ontologies/universita>
WHERE { ?stud un:nome "Mario" .
        ?stud un:cognome "Rossi" .
        ?stud un:esami_dati ?exam .
        ?exam un:corso_esame ?course .
        ?exam un:votazione ?vote .
        ?exam un:data ?date .
        ?course un:docente ?teacher .
        ?teacher un:nome "Maria Teresa" .
        ?teacher un:cognome "Pazienza" .
 }
ORDER BY ?date
```

Esercizio 9

Con riferimento alla seguente ontologia, sapendo che una persona ***deve*** pagare una multa se:
è un passeggero *e* non appartiene ad alcuna categoria speciale *e* non ha pagato il biglietto

Quesito 1:
È possibile rappresentare una classe in OWL (ovvero in DL) che rappresenti tutte le persone per le quali è possibile provare che non devono pagare una multa?

Quesito 2:
Sotto l'ipotesi che ogni persona sia dichiarata come appartenente al più ad una classe, realizzare una una query SPARQL per stabilire chi debba pagare una multa. Si vuole considerare una base di conoscenza chiusa e quindi ragionare in termini di Closed World Assumption

Quesito 3:
Fornire la lista delle istanze che risolvono il quesito 2, motivando la soluzione caso per caso.

Classi

```
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="Viaggio"/>
<owl:Class rdf:ID="Passeggero">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#passeggero_su"/>
            <owl:someValuesFrom rdf:resource="#Viaggio"/>
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="CategoriaSpeciale">
    <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="PortatoreDiHandicap">
    <rdfs:subClassOf rdf:resource="#CategoriaSpeciale"/>
</owl:Class>
<owl:Class rdf:ID="LavoratoreFerrovie">
    <rdfs:subClassOf rdf:resource="#CategoriaSpeciale"/>
</owl:Class>
<owl:Class rdf:ID="FiglioDiLavoratoreFerrovie">
    <owl:equivalentClass>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#madre"/>
                    <owl:someValuesFrom rdf:resource="#LavoratoreFerrovie"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#padre"/>
                    <owl:someValuesFrom rdf:resource="#LavoratoreFerrovie"/>
                </owl:Restriction>
            </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#CategoriaSpeciale"/>
</owl:Class>
```

Proprietà

```
<owl:ObjectProperty rdf:ID="viaggiatori">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
    <owl:inverseOf>
        <owl:FunctionalProperty rdf:about="#passeggero_su"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Viaggio"/>
    <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="eta">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="padre">
    <rdfs:range rdf:resource="#Person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="madre">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#Person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#passeggero_su">
    <owl:inverseOf rdf:resource="#viaggiatori"/>
    <rdfs:range rdf:resource="#Viaggio"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="dotato_di_biglietto">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdfs:domain rdf:resource="#Person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
```

Istanze

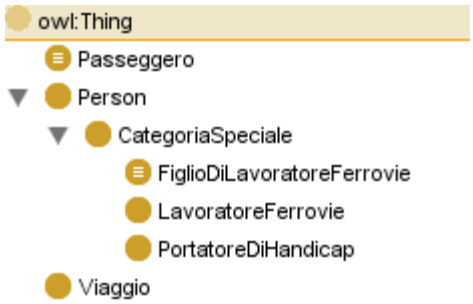
```
<Viaggio rdf:ID="Roma-Milano">
    <viaggiatori rdf:resource="#Simona"/>
    <viaggiatori rdf:resource="#Salvatore"/>
    <viaggiatori rdf:resource="#Pino"/>
    <viaggiatori rdf:resource="#Matilde"/>
    <viaggiatori rdf:resource="#Mario"/>
    <viaggiatori rdf:resource="#Mariolina"/>
    <viaggiatori rdf:resource="#Arcimboldo"/>
</Viaggio>


<Person rdf:ID="Simona">
        <passeggero_su rdf:resource="#Roma-Milano"/>
        <dotato_di_biglietto rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
        </dotato_di_biglietto>
    </Person>
    <Person rdf:ID="Arcimboldo">
        <passeggero_su rdf:resource="#Roma-Milano"/>
    </Person>
    <Person rdf:ID="Salvatore">
        <madre rdf:resource="#Mariolina"/>
        <dotato_di_biglietto rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
        </dotato_di_biglietto>
    </Person>
    <Person rdf:ID="Umberto"/>
    <Person rdf:ID="Annibale">
        <eta rdf:datatype="http://www.w3.org/2001/XMLSchema#int">42</eta>
        <dotato_di_biglietto rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        true
        </dotato_di_biglietto>
        <passeggero_su rdf:resource="#Roma-Milano"/>
    </Person>
    <Person rdf:ID="Mario">
        <dotato_di_biglietto rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
        </dotato_di_biglietto>
        <padre rdf:resource="#Pino"/>
    </Person>
    <LavoratoreFerrovie rdf:ID="Mariolina">
        <dotato_di_biglietto rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
        </dotato_di_biglietto>
    </LavoratoreFerrovie>
    <LavoratoreFerrovie rdf:ID="Pino">
        <dotato_di_biglietto rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
        </dotato_di_biglietto>
    </LavoratoreFerrovie>
    <PortatoreDiHandicap rdf:ID="Matilde">
        <dotato_di_biglietto rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
        false
        </dotato_di_biglietto>
     </PortatoreDiHandicap>
```

Soluzione:

Innanzitutto, conviene leggere con calma il file owl/xml e provvedere ad una descrizione più compatta delle informazioni, di modo da facilitare il ragionamento in un secondo momento.
In particolare, può essere utile avere:

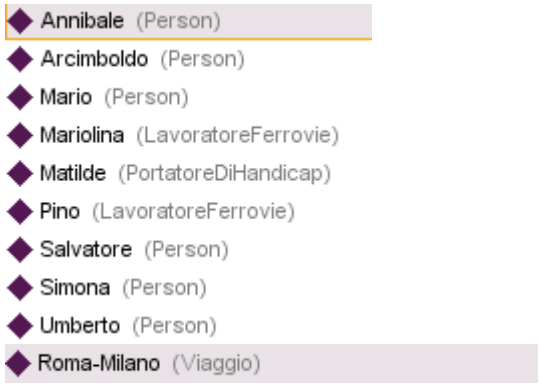
- Una visione ad albero della relazione IS-A esplicita tra le classi
- Un elenco delle istanze (individui) presenti nell'ontologia, riportante l'appartenenza esplicita di ogni individuo alle varie classi.
- Un elenco delle proprietà e delle loro "facets"
- Eventuali condizioni di equivalenza e/o subclass, che possono essere di aiuto durante la fase di reasoning.

Visione della tassonomia presente nell'ontologia:



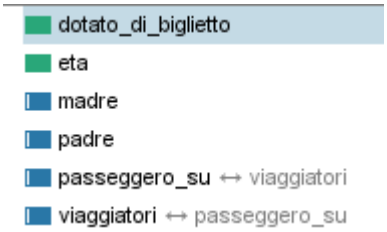
Ho evidenziato con il segno ≡ le classi per le quali sono definite delle relazioni di equivalenza (vedi più avanti).

Elenco istanze e appartenenza esplicita alle classi:



Questo è l'elenco completo degli individui presenti nell'ontologia e delle classi dei quali sono istanze dirette.

Elenco delle proprietà:



In questo caso con il colore verde ho evidenziato le `DatatypeProperties` e con il colore azzurro le `ObjectProperties`.
Ho anche segnalato la relazione di inverseOf tra `passeggero_su` e `viaggiatori`.

Condizioni asserite:

Utilizzando la notazione originale delle Description Logics, rappresento in forma compatta alcune informazioni sugli elementi della mia ontologia:

Passeggero ≡ ∃ passeggero_su.Viaggio

> La classe Passeggero è equivalente alla classe degli individui che hanno almeno una relazione di tipo passeggero_su con una istanza della classe Viaggio.

FiglioDiLavoratoreFerrovie ≡ ∃ madre.LavoratoreFerrovie ⊔ ∃ padre.LavoratoreFerrovie

> La classe FiglioDiLavoratoreFerrovie è equivalente alla classe degli individui che hanno padre e/o madre lavoranti alle ferrovie.

Quesito 1: Studio del predicato

Sapendo le condizioni affinché una persona debba pagare una multa, e cioè:

multato = "è un passeggero *e* non appartiene ad alcuna categoria speciale *e* non ha pagato il biglietto"

posso stabilire l'espressione per calcolare chi *sicuramente* non deve pagare una multa.

applico quindi De Morgan e derivo che:

non multato = *non* è un passeggero *o* appartiene ad una categoria speciale *o* ha pagato il biglietto.

Occorre ora ricordarci di una delle due assunzioni fondamentali di OWL:

OWA: Open World Assumption

In ipotesi di mondo chiuso, si ha la cosiddetta: negation-as-failure (NF): "se non riesco a dimostrare una cosa, allora questa è falsa."

In ipotesi di open world, la mancanza di informazione causa una impossibilità di decidere. Lo stato di ogni individuo rispetto ad una classe può quindi assumere tre valori:

- appartiene
- non appartiene
- non so

Per questo motivo posso stabilire l'insieme delle persone che *sicuramente* non devono essere multate, ma non sono in grado di stabilire se questo insieme risulta completo.
Nella nostra ontologia, in particolare, non tutte le persone hanno una madre e/o un padre specificati. Per quel che ne sappiamo, potrebbero essere figli di dipendenti delle ferrovie e quindi esenti da pagamento del biglietto, ma noi non abbiamo elementi per deciderlo.

Passeggeri o meno?

Umberto non risulta sul treno Roma-Milano ma, a meno di non avere altri vincoli che contraddicano questa affermazione, potrebbe anche salito su un altro treno (o su questo stesso!).
La presenza del vincolo di *proprietà funzionale* sulla proprietà passeggero_su mi garantisce che una persona non può essere su più di un viaggio.
Se avessi aggiunto l'istanza "a casa" sotto la classe Viaggio, avrei avuto un mezzo per indicare la presenza esplicita di "non passeggeri".

Provvisti di biglietto o no?

Se l'informazione manca del tutto, non abbiamo i mezzi per affermare se una persona è o meno sprovvista di biglietto.

Tornando alla espressione utilizzata per trovare il nostro insieme obiettivo:

non multato = *non* è un passeggero *o* appartiene ad una categoria speciale *o* ha pagato il biglietto.

possiamo escludere la prima condizione in quanto non è possibile provare quali persone *non sono* passeggeri e limitarci a cercare chi appartiene ad una categoria speciale *o* ha pagato il biglietto.

Possiamo creare una classe in equivalenza con le condizioni precedentemente descritte, quindi:

NonMultato = ¬Passeggero ⊔ CategoriaSpeciale ⊔ dotato_di_biglietto ∋∋True.

Quesito 2: Studio della query

Sapendo le condizioni affinché una persona debba pagare una multa, e cioè:

multato = "è un passeggero *e* non appartiene ad alcuna categoria speciale *e* non ha pagato il biglietto"

per sapere chi deve sicuramente pagare una multa, possiamo svolgere la query SPARQL per le singole parti e poi fare l'AND di esse.

Passeggero: questa classe è stata definita da noi. Dato che i sistemi per l'accesso a RDF possono essere impilati uno sull'altro, se il nostro RDF server fosse dotato di un reasoner OWL, potremmo tranquillamente considerare la tripla:

```
?person a un:Passeggero .
```

nella WHERE.

altrimenti, possiamo scrivere direttamente la clausola:

```
?person  mul:passeggero_su _:o .
```

Avendo inserito un blank_node (nodo anonimo) sull'oggetto. In realtà, in OWA, anche Umberto potrebbe essere un passeggero, del quale non è riportato lo stato sul presente documento, ma noi assumiamo di avere tutta l'informazione e di poter lavorare in CWA.

Non ha pagato il biglietto: in questo caso possiamo utilizzare un pattern molto comune per implementare la CWA, ossia una clausola OPTIONAL seguita dal controllo di UNBOUND (vedi slide su SPARQL).

```
OPTIONAL {?person mul:dotato_di_biglietto ?ticket .}
FILTER( !bound(?ticket) || (?ticket = false ) ) .
```

la prima clausola recupera il valore della prop dotato_di_biglietto, tenendo però, grazie all'OPTIONAL, tutte le istanze, comprese quindi di Arcimboldo, che non ha un valore sul biglietto (non si saprebbe quindi, in OWA, se lo ha o meno, mentre in CWA assumiamo che se non è specificato, allora è sprovvisto di biglietto).
Considerando quindi l'assenza di informazione come informazione negativa, si filtra il caso della variabile ticket UNBOUND (ossia rimasto variabile e non istanziato con alcun valore) e la si unisce al caso, banale, in cui questa sia false (cioè è dimostrato che il passeggero non ha biglietto).

Non appartenente a categoria speciale: Qui la faccenda si complica. SPARQL non ha (al momento) degli operatori per effettuare una differenza tra insiemi[1] (che potremmo usare per sottrarre l'insieme delle persone da non multare, ottenute per inferenza sulle sottoclassi di PersonaDaNonMultare, da quelle passabili di multa grazie ai criteri precedenti). Nell'ipotesi specificata nell'esercizio, e cioè che ogni individuo ha una sola classe tipo, possiamo svolgere noi tale differenza, per ognuno dei sottocasi di categoria speciale.

```
PREFIX mul: <http://art.uniroma2.it/ontologies/exercises/multe#>
SELECT ?person ?type
FROM <http://art.uniroma2.it/ontologies/exercises/multe>
WHERE {

  //prendo tutti i passeggeri
  ?person  mul:passeggero_su _:o .

  //tutti quelli che non hanno pagato biglietto
  OPTIONAL {?person mul:dotato_di_biglietto ?ticket .}
  FILTER( !bound(?ticket) || (?ticket = false ) ) .

  //tutti quelli che non sono lavoratori delle ferrovie o che non sono portatori di handicap
  ?person a ?type
  FILTER(?type != mul:LavoratoreFerrovie && ?type != mul:PortatoreDiHandicap)
```

[1] In realtà, questa assenza è puramente sintattica, perché in [Renzo Angles and Claudio Gutierrez, The Expressive Power of SPARQL, 2008] si dimostra l'equivalenza tra SPARQL è l'algebra relazionale, comprendente l'operatore \. È pur vero che il processo di riscrittura per ottenere tale differenza è a volte oneroso, producendo delle query quasi illeggibili. Future versioni del linguaggio includeranno probabilmente tale operatore.

```
  //tutti quelli che non hanno genitori lavoratori delle ferrovie. È divisa principalmente nelle due
  //parti relative a madre (la prima) e padre (la seconda). Entrambe sono ottenute da una union
  //tra quelli che non hanno madre/padre presenti nella KB e quelli che hanno il padre/madre che
  //NON E' lavoratore alle ferrovie
  {
    { OPTIONAL {?person mul:madre ?mother .}
      FILTER( !bound(?mother) )
    }
     UNION
    {
      ?person mul:madre ?mother .
      ?mother a ?mumtype
      FILTER(?mumtype != mul:LavoratoreFerrovie)
    }
  }

  {
    { OPTIONAL {?person mul:padre ?father .}
      FILTER( !bound(?father) )
    }
     UNION
    {
      ?person mul:padre ?father.
      ?father a ?fattype
      FILTER(?fattype != mul:LavoratoreFerrovie)
    }
  }
}
ORDER BY ?person
```

Notare che rimuovendo l'ipotesi di "una sola classe/tipo per ogni individuo" questa query non è più efficace, in quanto, per ogni clausola del tipo:

```
?person a ?type
```

vengono effettuati più binding. In questo modo, una persona che appartiene ad una CategoriaSpeciale, avrà un binding rimosso, perché non soddisfa la condizione

```
FILTER(?type != mul:LavoratoreFerrovie && ?type != mul:PortatoreDiHandicap)
```

e gli altri, corrispondenti ad altre classi di appartenenza compatibili con quella condizione, presi.

Quesito 3: istanze multate

Simona viaggia sul Roma-Milano, non appartiene (ne' ha genitori appartenenti) ad alcuna categoria speciale e ha un valore false su dotato_di_biglietto. Arcimboldo stessa cosa, ma non ha un valore su dotato_di_biglietto. Per CWA è cmq considerato sprovvisto di biglietto.