Assignment 2: Ontology Modeling

Lorenzo Lorgna, Data Semantics, Università degli Studi di Milano Bicocca 01/09/2021

Richiesta

Modify the **companies-geo.rdfs-owl-L2-2.owl** ontology with Protégé by creating two ontologies as specified below.

1. companies-geo.rdfs-owl-L2-2-Your_Surname_clean.owl

These specifications must be satisfied in ensemble in the final ontology; the order in which they are listed here below does not need to reflect the optimal order for a sequential implementation.

- Add two more branches to the ontology (the new classes are highlighted in bold):
  - **Person** > **Student**
  - Organization > **University**
- Make sure that each new class has at least two instances; you must appear as an instance of Student
- Model the following patterns by specifying the required object and datatype properties and by introducing two triples for each the pattern
  - Persons are friends to each other
  - Persons have a birthplace.
  - Persons have names
  - Students attends universities
  - Universities have addresses
- Interpret the above patterns by specifying domain and range restrictions over object and datatype properties so as to implement intuitively sound inferences in the ontology
- Use a reasoner to make sure that there are at least two examples of desired inferences, based on your axiomatization

Provided that you satisfy the above specifications, you can make any additional change to the ontology needed to improve your ontology.

Remarks:

- At the exam, you may be asked questions about your ontology, e.g., to justify your choices

2. companies-geo.rdfs-owl-L2-2-Your_Surname_wrong.owl

Modify the **companies-geo.rdfs-owl-L2-2-Your_Surname_clean.owl** ontology in such a way that either an inconsistency is derived or an unintended (or, counterintuitive, or undesirable) inference is drawn.

Risoluzione

Per la risoluzione è stato utilizzato l'editor di ontologie Protégé.

1. companies-geo.rdfs-owl-L2-2-Your_Lorgna_clean.owl

Come da richiesta, a partire dall'ontologia fornita, sono state aggiunte le seguenti componenti:

- Persona e Studente, quest'ultima come sottoclasse della prima
- University, come sottoclasse di Organization

Quest'operazione è stata eseguita navigando nella tab *Classes*, facente parte di *Entities*. A partire da *owl:Thing* utilizzando la funzione *add subclass* sono state aggiunte le due componenti sopra definite.

A seguire sono state create delle istanze di tali nuove classi, come ad esempio: *Alessandro è una persona*, *Lorenzo è uno studente*, *Bicocca è un'università*.

Come passo successivo sono stati modellati i pattern richiesti a partire dal tab *Object* properties, utilizzando la funzione *add a sub property*.

Inoltre per i pattern sopra definiti sono stati specificati anche *domain* e *range*, oltre che la tipologia di relazione (funzionale, transitiva ecc.)

A seguire sono stati definite almeno 2 triple per i pattern in questione come ad esempio: Alessia è nata a Milano, Andrea è amico di Alessandro, Lorenzo frequenta la Bicocca.

Viene utilizzato il reasoner per controllare la presenza o meno delle inferenze desiderate:

- Riccardo è amico di Lorenzo, dunque risulta correttamente che Lorenzo è a sua volta amico di Riccardo (proprietà riflessiva)
- Pur non avendo definito il tipo dell'entità Riccardo quest'ultimo viene definito correttamente come studente dal momento in cui è stato dichiarato che Riccardo frequenta la Bicocca (il dominio della funzione frequenta è studenti)

In aggiunta alle richieste sono state apportare ulteriori modifiche quali:

- nel tab individuals settare per le città il campo "different individuals" in modo tale che una persona non può essere nata a Milan e a Berlin. Se non viene apportata tale modifica, essendo la relazione wasBornIn funzionale, le due città vengono interpretate come la stessa entità

2. companies-geo.rdfs-owl-L2-2-Your_Surname_wrong.owl

Viene modificata l'ontologia sopra creata per far emergere delle inconsistenze. Ad esempio:

- *una persona non può essere nata in due città*: si aggiunge che Lorenzo è nato a Berlino, si genera correttamente un'inconsistenza (l'object property wasBornIn è funzionale).
- *una persona non può avere come luogo di nascita un'università*: si aggiunge che Lorenzo è nato in Bicocca, si genera correttamente un'inconsistenza (l'object property wasBornIn ha come range City).

- *una persona non può essere amica di se stessa*: si aggiunge che Alice è amica di se stessa, si genera correttamente un'inconsistenza (l'object property isFriendOf è simmetrica ma irriflessiva).
- *una persona non può frequentare una città*: si aggiunge che Lorenzo frequenta università Berlino, si genera correttamente un'inconsistenza (l'object property attendUniversity ha come range University).
- *un'università non può avere più indirizzi*: si aggiunge che Bicocca ha come ulteriore indirizzo Via XXV Aprile, si genera correttamente un'inconsistenza (la data property address è funzionale).

Vedi le immagini per spiegazione inconsistenze.