Reasoning su ontologie

Dato il seguente frammento di ontologia

```
<owl:Class rdf:ID="Persona"/>
    <owl:Class rdf:ID="Uomo">
        <rdfs:subClassOf rdf:resource="#Persona"/>
    </owl:Class>
    <owl:Class rdf:ID="Donna">
        <rdfs:subClassOf rdf:resource="#Persona"/>
    </owl:Class>
    <owl:Class rdf:ID="Simbolo"/>
    <owl:Class rdf:ID="Stark">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#marito"/>
                        <owl:someValuesFrom rdf:resource="#Stark"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#padre"/>
                        <owl:someValuesFrom rdf:resource="#Stark"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#emblema"/>
                        <owl:hasValue rdf:resource="#Lupo"/>
                    </owl:Restriction>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>
    <owl:Class rdf:ID="Lannister">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#marito"/>
                        <owl:someValuesFrom rdf:resource="#Lannister"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#padre"/>
                        <owl:someValuesFrom rdf:resource="#Lannister"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#emblema"/>
                        <owl:hasValue rdf:resource="#Leone"/>
                    </owl:Restriction>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>
    <owl:ObjectProperty rdf:about="#padre">
        <rdfs:domain rdf:resource="#Persona"/>
        <rdfs:range rdf:resource="#Uomo"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    </owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#emblema">
    <rdfs:domain rdf:resource="#Persona"/>
    <rdfs:range rdf:resource="#Simbolo"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="marito">
    <rdfs:domain rdf:resource="#Donna"/>
    <rdfs:range rdf:resource="#Uomo"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<Uomo rdf:ID="Tyrion">
    <emblema rdf:resource="#Leone"/>
</Uomo>
<owl:Thing rdf:ID="Sansa">
    <emblema rdf:resource="#Lupo"/>
    <marito rdf:resource="#Tyrion"/>
</owl:Thing>
<Simbolo rdf:ID="Lupo"/>
<Simbolo rdf:ID="Leone"/>
```

Determinare:

1. eventuali inconsistenze
2. Per ognuna delle classi dichiarate, le relative istanze (esplicite o inferibili)
3. Per ogni classe, eventuali superclassi inferibili

Motivate il vostro ragionamento

Soluzione:

1) non ce ne sono. Sansa può appartenere tranquillamente sia alla casa Stark (per emblema) che alla casa Lannister (per marito Tyrion che ha il Leone come emblema).

2) Lannister: Sansa e Tyrion (vedi sopra)
   Stark: Sansa (ha emblema Lupo)
   Donna: Sansa (ha un marito)

3) Per Lannister e Stark: Persona (perché sono union di classi che hanno sempre almeno un valore su una proprietà che ha come dominio persona, quindi in tutti i casi ho cmq una persona nel dominio)

Modellazione e Interrogazione

Data la seguente ontologia, che descrive il dominio della formula1:

```
<owl:Class rdf:ID="Scuderia"/>
<owl:Class rdf:ID="GP"/>
<owl:Class rdf:ID="Pilota"/>
<owl:Class rdf:ID="WorldZone"/>
<owl:ObjectProperty rdf:ID="arrivato_secondo">
    <owl:inverseOf>
        <owl:FunctionalProperty rdf:ID="secondo_classificato"/>
    </owl:inverseOf>
    <rdfs:subPropertyOf>
        <owl:ObjectProperty rdf:ID="arrivo"/>
    </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="arrivato_primo">
    <owl:inverseOf>
        <owl:FunctionalProperty rdf:ID="primo_classificato"/>
    </owl:inverseOf>
    <rdfs:subPropertyOf>
        <owl:ObjectProperty rdf:about="#arrivo"/>
    </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#location">
    <rdfs:domain rdf:resource="#GP"/>
    <rdfs:range rdf:resource="#WorldZone"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#arrivo">
    <rdfs:domain rdf:resource="#Pilota"/>
    <rdfs:range rdf:resource="#GP"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="classificato">
    <rdfs:domain rdf:resource="#GP"/>
    <rdfs:range rdf:resource="#Pilota"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="arrivato_terzo">
    <owl:inverseOf>
        <owl:FunctionalProperty rdf:ID="terzo_classificato"/>
      </owl:inverseOf>
    <rdfs:subPropertyOf rdf:resource="#arrivo"/>
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:about="#primo_classificato">
    <rdfs:subPropertyOf rdf:resource="#classificato"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#secondo_classificato">
    <rdfs:subPropertyOf rdf:resource="#classificato"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#terzo_classificato">
    <rdfs:subPropertyOf rdf:resource="#classificato"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="corre_per">
    <rdfs:range rdf:resource="#Scuderia"/>
    <rdfs:domain rdf:resource="#Pilota"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
```

```
<Pilota rdf:ID="nullPilot">
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">questo è un pilota vuoto da assegnare alle proprietà di tipo "classificato" quando nessuno è riuscito a classificarsi in quella posizione</rdfs:comment>
</Pilota>
<WorldZone rdf:ID="europe"/>
<WorldZone rdf:ID="asia"/>
<WorldZone rdf:ID="africa"/>
```

D1:

definire attraverso una opportuna espressione in Description Logics:

- tutti i piloti che hanno vinto (primi classificati) almeno un GP in una città europea
- tutti i GP in cui sono arrivate non più di due macchine al traguardo

R1:

1. EuropeWinner ≡ ∃ arrivato_primo (location ∋europe)
2. GPArriviIncompleti ≡ terzo_classificato ∋ nullPilot

D2:

Definire una query SPARQL che, dato un pilota (ad es. palladini, supponete di avere tale istanza nell'ontologia), ottenga tutti i suoi compagni di scuderia

R2:

```
SELECT ?compagno
WHERE {
 :palladini  :corre_per   ?scuderia  .
 ?compagno   :corre_per   ?scuderia  .
 FILTER  (?compagno != :palladini)    .  //per eliminare lui stesso dai risultati
}
```

D3:

Completando eventualmente la terminology box dell'ontologia, è possibile ottenere gli stessi risultati della SPARQL query precedente attraverso una descrizione ontologica?

R3:

Introducendo la proprietà: piloti, inversa di corre_per, potremo definire:

CompagnoPalladini ≡ ∃ corre_per (piloti ∋ palladini)

D4:

Viceversa, è possibile ottenere, attraverso una query SPARQL, gli stessi risultati del secondo concetto richiesto nella domanda D1, qualora la presenza di arrivi vuoti su determinate posizioni non sia però qualificata da un nullPilot?
Se sì, motivate il vostro ragionamento

R4:

Come già detto a lezione, ciò richiede una negation-as-failure, concetto in genere accompagnato all'ipotesi di closed-world-assumption. Per questo motivo, invece di lasciare vuoti gli slot relativi

agli arrivi, si è introdotto nella ontologia un esplicito pilota nullPilot su cui poter effettuare delle ricerche in modo "affermativo".
È pur vero che SPARQL (vedi slide) ammette la possibilità di implementare il NOT in taluni casi (e quindi la negation-as-failure), tramite uso opportuno di BOUND e OPTIONAL, per cui sarebbe in grado di catturare dei GP su cui semplicemente la proprietà terzo_classificato non ha alcun valore.

Di seguito la query:

```
SELECT ?gp ?pilota
WHERE  {
  ?gp  a   :GP .
  OPTIONAL { ?gp :terzo_classificato ?pilota . }
  FILTER( !bound( ?pilota ) )
}
```