

For return on 20 January 2012 (late submission: 3 February 2012)

*Electronic submission: .pdf and .owl files only*

1. (8%) Consider the following RDF document:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:lit="http://literature.org/#"
  xml:base="http://literature.org/">

  <rdf:Description rdf:about="#Hamlet">
    <rdf:type rdf:resource="#drama"/>
  </rdf:Description>

  <rdf:Description rdf:about="#Sonet96">
    <rdf:type rdf:resource="#poem"/>
  </rdf:Description>

  <rdf:Description rdf:about="#wrote">
    <rdf:type rdf:resource="#rdf:Property"/>
    <rdf:domain rdf:resource="#writer"/>
    <rdf:range rdf:resource="#literary_content"/>
  </rdf:Description>

  <rdfs:Class rdf:about="#poem"/>
  <rdfs:Class rdf:about="#drama"/>

  <lit:poet rdf:about="#Shakespear">
    <lit:wrote rdf:resource="#Sonet96"/>
    <lit:wrote rdf:resource="#Hamlet"/>
    <rdf:type rdf:resource="#playwright"/>
  </lit:poet>

</rdf:RDF>
```

- (a) Describe in natural language the content of this document.
- (b) Draw the graph representation of the document.
- (c) Explain which of the following can or cannot be represented in RDF(S):
  - Poets and playwrights are writers.
  - Poets write poems; playwrights write dramas.
  - Poets do not write RDF documents.

**Answer.** (a) Hamlet is a drama. Sonet 96 is a poem. Shakespear is a poet who wrote Sonet 96 and Hamlet; he is also a playwright. Here, ‘wrote’ is a property with domain ‘writer’ and range ‘literary content’; ‘poem’, ‘drama’, ‘writer’ and ‘literary content’ are classes.

(b) The corresponding RDF(S) triples are as in lecture slides.

(c) ‘Poets and playwrights are writers’ can be represented in RDF(S) as two triples: ‘poet is a writer’ and ‘playwright is a writer.’ ‘Poets write poems; playwrights write dramas’ cannot be represented in RDF(S) because the property ‘write’ can have only one domain and one range. If we state that the range of ‘write’ is ‘Poem’ and ‘Drama’ then this would mean that the range is actually the intersection of the two. RDF(S) does not support localised domain and range constraints. A description logic representation:

$$\text{Poet} \sqsubseteq \exists \text{write.Poem}, \quad \text{Playwright} \sqsubseteq \exists \text{write.Drama}.$$

**2. (10%)** Represent the following information by means of RDF/S triples:

- a person, whose emails address is `bob@somewhere.uk`, plays golf with the spouse of a colleague who has two email addresses: `rob@elsewhere.uk` and `rob@somewhere.it`.

(Hint: use blank nodes as on pages 14–16, Lecture 3. You can use the Turtle syntax.) Draw the graph representation of the RDF/S document. Represent the same information using the language of description logic.

**Answer.** Recall that a blank node in RDF is a node in an RDF graph representing a resource for which a URI or literal is not given. The resource represented by a blank node is also called an anonymous resource. The following is a Turtle representation:

```
_:p1 rdf:type foaf:Person
_:p2 rdf:type foaf:Person
_:p3 rdf:type foaf:Person
_:p1 foaf:hasEmail ex:bob@somewhere.uk
_:p1 foaf:playsGolfWith _:p2
_:p2 foaf:hasSpouse _:p3
_:p1 foaf:hasColleague _:p3
_:p3 foaf:hasEmail ex:rob@somewhere.uk
_:p3 foaf:hasEmail ex:rob@elsewhere.it
```

Here `_:p1`, `_:p2` and `_:p3` are three blank nodes for the three persons involved in the text. The graph representation for these triples is obvious.

Individuals in the OWL 2 syntax represent actual objects from the domain. There are two types of individuals in the syntax of OWL 2. Named individuals are given an explicit name that can be used in any ontology to refer to the same object. Anonymous individuals do not have a global name and are thus local to the ontology they are contained in. Thus, we can represent the same information as part of an ABox:

```
(_:p1,bob@somewhere.uk) : hasEmail
( _:p1, _:p2 ) : playsGolfWith
( _:p2, _:p3 ) : hasSpouse
( _:p1, _:p3 ) : hasColleague
( _:p3,rob@somewhere.uk ) : hasEmail
( _:p3,rob@elsewhere.it ) : hasEmail
```

3. (10%) Consider the following data and background knowledge:

- (1) John manages 'Eden'.
  - (2) Persons who manage projects are managers.
  - (3) One can only manage projects or departments.
  - (4) All managers manage something.
  - (5) Managers can be either top managers or area managers.
  - (6) Every employee has a manager and works on a project.
  - (7) One can only work on a project.
  - (8) Area managers manage projects.
  - (9) Those who have managers are employees.
  - (10) John is not a top manager.
- (a) Which of the above statements can be satisfactorily modelled in RDF/S? Explain your answer and give the corresponding RDF/S graph representations.
  - (b) Represent the statements above as a description logic knowledge base, indicating the TBox and the ABox.
  - (c) What is the answer to the query 'is Eden a project?' with respect to the resulting knowledge base.

**Answer.** (a) Statement (1) can be represented by the RDF triple (*john*, *manages*, *eden*). (2) can be modelled by saying the domain of *manages* is *manager*, but this is not perfect. (3) requires the Boolean 'or' which is not available in RDF/S. (4) In RDFS, one can say that *manager* is a subclass of the domain of *manages*. But this is not satisfactory. (5) needs 'or'. (6) *employee* is a subclass of the domain of *has* and the domain of *worksOn*. (7) the range *worksOn* is *project*. (8) there is no universal quantification in RDF/S. (9) the domain of *has* is a subclass of *employee*; not satisfactory. (10) no negation in RDF/S.

(b) An *ALCI* representation:

- (1) (*john*, *eden*) : *manages*
- (2)  $\text{Person} \sqcap \exists \text{manages.Project} \sqsubseteq \text{Manager}$
- (3)  $\exists \text{manages}^{\neg} . \top \sqsubseteq \text{Project} \sqcup \text{Department}$
- (4)  $\text{Manager} \sqsubseteq \exists \text{manages} . \top$
- (5)  $\text{Manager} \sqsubseteq \text{TopManager} \sqcup \text{AreaManager}$ ,  $\text{TopManager} \sqcap \text{AreaManager} \sqsubseteq \perp$
- (6)  $\text{Employee} \sqsubseteq \exists \text{has.Manager} \sqcap \exists \text{worksOn.Project}$
- (7)  $\exists \text{worksOn}^{\neg} . \top \sqsubseteq \text{Project}$
- (8)  $\text{AreaManager} \sqsubseteq \exists \text{manages.Project} \sqcap \forall \text{manages.Project}$
- (9)  $\exists \text{has.Manager} \sqsubseteq \text{Employee}$
- (10)  $\text{john} : \neg \text{TopManager}$

(c) The answer is ‘yes’. Indeed, Eden is either a project or a department. John is not a top manager, so he must be an area manager, who can only manage projects. So Eden is a project.

4. (16%) Sketch a normalised *movie ontology*, which covers the items listed below and provides sufficient concepts and roles for part (b).

- (a) Indicate the hierarchies for both concepts (classes) and properties (roles). For concepts, indicate clearly which are self-standing, which are definable and which are modifiers. Introduce instances of concepts if required. For properties, define their domains and ranges, their inverses, and indicate whether the roles are (ir)reflexive, (a)symmetric, functional, inverse-functional or transitive.

Items to be represented: Genre, Award, Actor, Production Company, Action, Comedy, Director, Oscar, Hollywood.

- (b) Define classes using your ontology for the items below (using the OWL functional-style syntax or a reasonable approximation) or explain why they cannot be expressed in OWL:

- Award winning comedy actor.
- Hollywood movie made in 1961.
- Company producing comedies and documentaries only.

If the definition in English is ambiguous, paraphrase it so it is unambiguous and then express the disambiguated notion in OWL.

**Answer.** (a) A normalised ontology:

self-standing	relations	definables	modifiers
Person Movie Award – Oscar ProductionCompany Location – Hollywood – Bollywood Year – 1961 (individual)	won (range: Award) madeIn (func.) locatedIn (func.) playsIn (range: Movie) rankedHigherThan (asym., irref.)	Actor Director	Genre – Action – Comedy – Documentary

- EquivalentClasses(AwardWinningComedyActor  
 ObjectIntersectionOf  
 (Actor  
 ObjectSomeValuesFrom(won Award)  
 ObjectSomeValuesFrom(playsIn

ObjectSomeValuesFrom(hasGenre Comedy))))))

In the DL syntax:

$\text{AwardWinningComedyActor} \equiv \text{Actor} \sqcap \exists \text{won.Award} \sqcap \exists \text{playsIn.}(\exists \text{hasGenre.Comedy})$

- $\text{EquivalentClasses}(\text{HollywoodMovieMadeIn1961} \text{ ObjectIntersectionOf}(\text{Movie} \text{ ObjectHasValue}(\text{madeIn } 1961) \text{ ObjectSomeValuesFrom}(\text{locatedIn Hollywood}))))))$
- $\text{EquivalentClasses}(\text{CompanyProducingComediesAndDocumetraies} \text{ ObjectIntersectionOf}(\text{ProductionCompany} \text{ ObjectSomeValuesFrom}(\text{ObjectUnionOf}(\text{Comedy Documentary})) \text{ ObjectAllValuesFrom}(\text{ObjectUnionOf}(\text{Comedy Documentary}))))))$

5. (10%) Consider the following small ontology written in OWL functional-style syntax:

$\text{DisjointClasses}(\text{Animal Plant})$   
 $\text{ObjectPropertyDomain}(\text{eats Animal})$   
 $\text{EquivalentClasses}(\text{Herbivore ObjectAllValuesFrom}(\text{eats Plant}))$   
 $\text{EquivalentClasses}(\text{Carnivore ObjectAllValuesFrom}(\text{eats Animal}))$   
 $\text{EquivalentClasses}(\text{CarnivorousPlant ObjectIntersectionOf}(\text{Plant Carnivore}))$

- Write down DL equivalents of each of the axioms in the ontology.
- Enter the above ontology into Protégé, and use the reasoner to compute the class hierarchy.
- Explain why Plant is a sub-class of both Herbivore and Carnivore.
- This does not seem to be correct. Explain how you would improve the ontology in order to fix this problem.
- Did your improvement reveal any other problem with the ontology? If so, how would you repair the problem?

**Answer:** DL equivalents of the axioms can be as follows:

- $\text{Animal} \sqsubseteq \neg \text{Plant}$
- $\exists \text{eats.T} \sqsubseteq \text{Animal}$
- $\text{Herbivore} \equiv \forall \text{eats.Plant}$
- $\text{Carnivore} \equiv \forall \text{eats.Animal}$
- $\text{CarnivorousPlant} \equiv \text{Plant} \sqcap \text{Carnivore}$

(c) As the classes `Plant` and `Animal` are disjoint and every object that eats something must be in the class `Animal`, the elements of `Plant` cannot eat anything, and so they satisfy the definitions of `Herbivore` and `Carnivore`.

(d) To fix the problem, one should state that each `Herbivore` eats some `Plant` (and nothing but `Plant`), and that each `Carnivore` eats some `Animal` (and nothing but `Animal`), that is,

- `Herbivore`  $\equiv \exists \text{eats.Plant} \sqcap \forall \text{eats.Plant}$
- `Carnivore`  $\equiv \exists \text{eats.Animal} \sqcap \forall \text{eats.Animal}$

(e) However, this makes `CarnivorousPlant` inconsistent because all objects that eat something must be `Animals`, and so not `Plants`. One solution could be to extend the domain of `eats` to `Plant`:

- $\exists \text{eats.T} \sqsubseteq \text{Animal} \sqcup \text{Plant}$

Or one can define a new class, say `LivingThing`, which contains both `Plant` and `Animal` and is the domain of the property `eats`.

**6. (10%)** Consider the description logic knowledge base  $\mathcal{K}$

`Student`  $\sqsubseteq \exists \text{hasTutor.Professor}$   
`Professor`  $\sqsubseteq (\exists \text{teaches.Student}) \sqcap (\neg \forall \text{teaches.Student})$   
`joe` : `Student`  
`sam` : `Student`  
`(sam, john)` : `hasTutor`

and the interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where

$\Delta^{\mathcal{I}} = \{a, b, c, d\}$ ,  $\text{joe}^{\mathcal{I}} = a$ ,  $\text{sam}^{\mathcal{I}} = b$ ,  $\text{john}^{\mathcal{I}} = c$ ,  
 $\text{Student}^{\mathcal{I}} = \{a, b\}$ ,  $\text{Professor}^{\mathcal{I}} = \{c, d\}$ ,  
 $\text{hasTutor}^{\mathcal{I}} = \{(a, c), (b, c)\}$ ,  $\text{teaches}^{\mathcal{I}} = \{(c, b), (c, d), (d, b)\}$ .

- (a) Is  $\mathcal{I}$  a model of  $\mathcal{K}$ ? Explain your answer by computing the extensions  $C^{\mathcal{I}}$  of all the concepts  $C$  from  $\mathcal{K}$ . Does  $\mathcal{I}$  satisfy the unique name assumption?
- (b) Is `john` an instance of `Professor` with respect to  $\mathcal{K}$ ? Explain your answer.
- (c) Is the concept `Student`  $\sqcap \exists \text{hasTutor}.\forall \text{teaches.Student}$  satisfiable with respect to  $\mathcal{K}$ ? Explain your answer.
- (d) Extend  $\mathcal{K}$  with `Student`  $\equiv$  `Professor`. Is the resulting knowledge base consistent? Explain your answer.
- (e) Assume that you have a reasoner that can check satisfiability of concepts with respect to a knowledge base. Can you use it to check that one concept is equivalent to another with respect to the knowledge base, for instance, whether `Student`  $\equiv$  `Professor`? Explain your answer.

**Answer.** (a) No,  $\mathcal{I}$  is not a model of  $\mathcal{K}$  because  $\text{Student}^{\mathcal{I}} \subseteq (\exists \text{hasTutor. Professor})^{\mathcal{I}}$ , but  $\text{Professor}^{\mathcal{I}} \not\subseteq (\exists \text{teaches. Student} \sqcap \neg \forall \text{teaches. Student})^{\mathcal{I}}$ , as  $d$  belongs to the left-hand side but not to the second concept of the right-hand side. As the interpretations of all the individual names are different,  $\mathcal{I}$  satisfies the unique name assumption.

(b) No,  $\mathcal{K} \not\models \text{john} : \text{Professor}$  because there is a model  $\mathcal{I}'$  of  $\mathcal{K}$  where  $\text{john}^{\mathcal{I}'} \notin \text{Professor}^{\mathcal{I}'}$ . For example, add  $c'$  to  $\mathcal{I}$ , set  $\text{john}^{\mathcal{I}'} = c'$  and extend  $\text{hasTutor}^{\mathcal{I}}$  by adding the pair  $(a, c')$  to it, and extend  $\text{teaches}^{\mathcal{I}}$  with the pair  $(d, c)$ .

(c) Yes, it is satisfiable. One can easily construct a model of  $\mathcal{K}$  containing a student with a tutor, who is not a professor and who teaches only students.

(d) Yes, it is. The following interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  is a model of  $\mathcal{K}$ :

$$\begin{aligned} \Delta^{\mathcal{I}} &= \{a, b, c\}, & \text{joe}^{\mathcal{I}} &= a, & \text{sam}^{\mathcal{I}} &= b, & \text{john}^{\mathcal{I}} &= c, \\ \text{Student}^{\mathcal{I}} &= \{a, b, c\}, & \text{Professor}^{\mathcal{I}} &= \{a, b, c\}, \\ \text{hasTutor}^{\mathcal{I}} &= \{(a, b), (b, c), (c, a)\}, & \text{teaches}^{\mathcal{I}} &= \{(b, a), (c, b), (a, c), (a, d), (b, d), (c, d)\}. \end{aligned}$$

(e)  $\mathcal{K} \models C \equiv D$  if and only if neither  $C \sqcap \neg D$  is satisfiable w.r.t.  $\mathcal{K}$ , nor  $D \sqcap \neg C$  is satisfiable w.r.t.  $\mathcal{K}$ .

**7. (9%)** Consider the following ABox  $\mathcal{A}$ :

$$\begin{array}{ll} (\text{ron}, \text{claudia}) : \text{likes} & (\text{ron}, \text{peter}) : \text{likes} \\ (\text{claudia}, \text{peter}) : \text{is\_neighbour\_of} & (\text{peter}, \text{andrea}) : \text{is\_neighbour\_of} \\ \text{claudia} : \text{Blond} & \text{andrea} : \neg \text{Blond} \end{array}$$

(a) Does  $\mathcal{A}$  have a model?

(b) Is  $\text{ron}$  an instance of the concept

$$\exists \text{likes.} (\text{Blond} \sqcap \exists \text{is\_neighbour\_of.} \neg \text{Blond})$$

in all models of  $\mathcal{A}$ ?

(c) Is  $\text{ron}$  an instance of the concept

$$\exists \text{likes.} (\exists \text{is\_neighbour\_of.} (\forall \text{is\_neighbour\_of.} \neg \text{Blond}))$$

in all models of  $\mathcal{A}$ ?

**Answer.** (a) The following interpretation  $\mathcal{I}$  is a model of  $\mathcal{A}$ :

$$\begin{aligned} \Delta^{\mathcal{I}} &= \{r, c, p, a\}, \\ \text{ron}^{\mathcal{I}} &= r, \\ \text{claudia}^{\mathcal{I}} &= c, \\ \text{peter}^{\mathcal{I}} &= p, \\ \text{andrea}^{\mathcal{I}} &= a, \\ \text{Blond}^{\mathcal{I}} &= \{c\}, \\ \text{likes}^{\mathcal{I}} &= \{(r, c), (r, p)\}, \\ \text{is\_neighbour\_of}^{\mathcal{I}} &= \{(c, p), (p, a)\} \end{aligned}$$

because it satisfies all the ABox assertions.

(b) Yes, it is. Let  $\mathcal{J}$  be a model of  $\mathcal{A}$ . Then we must have  $(r, c) \in \text{likes}^{\mathcal{J}}$  and  $c \in \text{Blond}^{\mathcal{J}}$ . So, if  $p \notin \text{Blond}$  then Ron is an instance of the concept in question. Suppose next that  $p \in \text{Blond}$ . But then Ron is again an instance of that concept because  $(r, p) \in \text{likes}^{\mathcal{J}}$ .

(c) No, this is not the case in the model obtained from  $\mathcal{I}$  above by adding two new elements  $x$  and  $y$  such that  $a, b \in \text{Blond}^{\mathcal{I}}$ ,  $(p, x) \in \text{is\_neighbour\_of}^{\mathcal{I}}$  and  $(a, y) \in \text{is\_neighbour\_of}^{\mathcal{I}}$ .

**8. (10%)** Consider the concept

$$C = \neg(\forall R. \neg A \sqcup \forall R. \neg(B \sqcap D)) \sqcap \neg \exists R. (A \sqcap B \sqcap D)$$

(i) Transform concept  $C$  to an equivalent concept in negation normal form.

(ii) Prove that concept  $C$  is satisfiable by giving a model of  $C$ . To construct the model, apply the  $\mathcal{ALC}$  tableau algorithm to the negation normal form of  $C$  obtained in (i).

**Answer.**  $C$  is equivalent to the following concept in NNF:

$$\exists R. A \sqcap \exists R. (B \sqcap D) \sqcap \forall R. (\neg A \sqcup \neg B \sqcup \neg D).$$

Applying the tableau algorithm to  $C$ :

$$\begin{aligned} S_0 &= \{x : \exists R. A, x : \exists R. (B \sqcap D), x : \forall R. (\neg A \sqcup \neg B \sqcup \neg D)\} \\ S_0 \rightarrow_{\exists} S_1 &= S_0 \cup \{(x, y) : R, y : A\} \quad (y \text{ is a fresh individual}) \\ S_1 \rightarrow_{\exists} S_2 &= S_1 \cup \{(x, z) : R, z : B \sqcap D\} \quad (z \text{ is a fresh individual}) \\ S_2 \rightarrow_{\sqcap} S_3 &= S_2 \cup \{z : B, z : D\} \\ + S_3 \rightarrow_{\sqcup} S_{4.1} &= S_3 \cup \{y : \neg B\} \\ S_{4.1} \rightarrow_{\sqcup} S_{5.1} &= S_{4.1} \cup \{z : \neg A\} \text{ — complete and clash-free} \end{aligned}$$

$S_{5.1}$  is a complete and clash-free constraint system. It induces the interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where

$$\Delta^{\mathcal{I}} = \{x, y, z\}, \quad R^{\mathcal{I}} = \{(x, y), (x, z)\}, \quad A^{\mathcal{I}} = \{y\}, \quad B^{\mathcal{I}} = D^{\mathcal{I}} = \{z\}.$$

**9. (10%)** Using the individuals john and helen, concept names PoorPerson, WealthyPerson, SmartPerson, PersonReadingNewspapers, HappyPerson, ExcitingSpouse and PersonWithExcitingLife and role hasSpouse, represent the following informal knowledge as an  $\mathcal{ALC}$  knowledge base:

- (1) People who are smart and not poor are happy.
- (2) People who read newspapers are smart.
- (3) John is wealthy.
- (4) Helen reads newspapers and is wealthy.
- (5) Happy people have exciting lives.
- (6) Wealthy people are not poor.
- (7) John has Helen as spouse.

(8) A spouse who is not exciting has only poor spouse.

(9) People who have a spouse with an exciting spouse are happy.

(a) Which of the statements above can be represented in RDF or RDFS?

(b) For each individual name, find the most specific (i.e., smallest) concepts containing it as an instance with respect to the knowledge base.

**Answer.**

(1)  $\neg\text{PoorPerson} \sqcap \text{SmartPerson} \sqsubseteq \text{HappyPerson}$

(2)  $\text{PersonThatReadsNewspapers} \sqsubseteq \text{SmartPerson}$

(3)  $\text{john} : \text{WealthyPerson}$

(4)  $\text{helen} : \text{PersonThatReadsNewspapers} \sqcap \text{WealthyPerson}$

(5)  $\text{HappyPerson} \sqsubseteq \text{PersonWithExcitingLife}$

(6)  $\text{WealthyPerson} \sqsubseteq \neg\text{PoorPerson}$

(7)  $(\text{john}, \text{helen}) : \text{hasSpouse}$  and  $(\text{helen}, \text{john}) : \text{hasSpouse}$

(8)  $\neg\text{ExcitingSpouse} \sqsubseteq \forall\text{hasSpouse}.\text{PoorPerson}$

(9)  $\exists\text{hasSpouse}.\exists\text{hasSpouse}.\text{ExcitingSpouse} \sqsubseteq \text{HappyPerson}$

(a) Statements (2), (3), (4), (5) and (7) can be represented as RDF/S triples. (1), (6) and (8) use negation, which is not available in RDF(S). (9) uses qualified existential restrictions, which are not available either.

(b) helen belongs to

- PersonThatReadsNewspaper and WealthyPerson (from (d)),
- SmartPerson (from (b)),
- HappyPerson (from (a)),
- PersonWithExcitingLife (from (e)).

john belongs to

- WealthyPerson (from (c)),
- ExcitingSpouse (from (h) and (g); otherwise helen would belong to PoorPerson, which is not the case because of (f) and (d)),
- HappyPerson (from (i) and (g) twice),
- PersonWithExcitingLife (from (e)).

Most specific concept names are underlined (other concept names in the lists above are subsumed by underlined ones).

Both helen and john are instances of PersonWithExcitingLife.

**10. (3%)** Explain how the notion of SymmetricProperty in OWL can be expressed in terms of other language primitives.

**Answer:** A property  $R$  is symmetric if, for all  $x, y$ , we have  $R(x, y)$  iff  $R(y, x)$ . In other words,  $R$  is symmetric iff  $R = R^-$ . This can be represented as

`ObjectProperty(R inverseOf(R))` in abstract OWL syntax and as  $R \equiv R^-$  in DL.

**11. (4%)** Write down one or more DL axioms that express the same constraints that are expressed in each of the following OWL axioms written in abstract syntax:

- `DisjointClasses(Meat Fish Vegetable Fruit)`
- `ObjectPropertyDomain(father Person)`
- `ObjectPropertyRange(father Male)`
- `SubObjectPropertyOf(father parent)`
- `FunctionalObjectProperty(father)`
- `SymmetricObjectProperty(hasSameGrade)`
- `SubClassOf(Lion ObjectIntersectionOf(Animal ObjectAllValuesFrom(eats Animal)))`
- `EquivalentClasses(Carnivore ObjectIntersectionOf(Animal ObjectSomeValuesFrom(eats Animal)))`

**Answer:**

- $\text{Meat} \sqsubseteq \neg\text{Fish}, \quad \text{Meat} \sqsubseteq \neg\text{Vegetable}, \quad \text{Meat} \sqsubseteq \neg\text{Fruit},$   
 $\text{Fish} \sqsubseteq \neg\text{Vegetable}, \quad \text{Fish} \sqsubseteq \neg\text{Fruit}, \quad \text{Vegetable} \sqsubseteq \neg\text{Fruit}$
- $\text{father} \sqsubseteq \text{parent}, \quad \top \sqsubseteq \leq 1\text{father}, \quad \exists\text{father}.\top \sqsubseteq \text{Person}, \quad \top \sqsubseteq \forall\text{father}.\text{Male}$
- $\text{hasSameGrade} \equiv \text{hasSameGrade}^-$
- $\text{Lion} \sqsubseteq \text{Animal} \sqcap \forall\text{eats}.\text{Animal}$
- $\text{Carnivore} \equiv \text{Animal} \sqcap \exists\text{eats}.\text{Animal}$