NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

Student's full-name: Spanos Nikolaos

Academic Number: 7115112100023

Teacher: Koumparakis Manolis

2nd academic homework exercise of course Knowledge Graphs

Academic year: 2022-2023

Master of Science in Computer Science

Athens, December 2022

Table of Contents

Question 1

Consider the following RDF graph G:

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix ex: <http://middle-earth.org/> .

ex:Thorin rdf:type ex:Dwarf .

ex:Thrain rdf:Type ex:Dwarf .

ex:Thorin ex:hasFather ex:Thrain .

ex:Thrain ex:hasFather ex:Thor .

ex:Thorin ex:hasTitle "King under the Mountain"

Express the query: Find all the dwarfs, their fathers and optionally their title using the algebraic formalism of the paper Jorge Perez, Marcelo Arenas, and Claudio Gutierrez. Semantics and Complexity of SPARQL. Proceedings of ISWC 2006.

Now define as D the dataset of this RDF:

$D = \{$

(N_1, rdf:type, ex:Dwarf), (N_1, ex:hasFather, ex:Thrain), (N_1, ex:hasTitle, "King under the Mountain")

(N_2, rdf:type, ex:Dwarf), (N_2, ex:hasFfather, ex:Thor),

(N_3, rdf:type, ex:Dwarf),

}

Note 1

For the shake of the solution and based on the knowledge graph $N_1 \dots N_3$ represent the entity ex:Thorin, ex:Thrain, ex:Thor respectively. If we had an expression like *rdf:name* we could add in the dataset D as define above, the following:

(N_1, rdf:name, ex:Thorin),

(N_2, rdf:name, ex:Thrain),

(N_3, rdf:name, ex:Thor),

But since the above is not known I didn't proceed with the addition explained above.

Note 2

Since ex:Thor is the father of ex:Thrain and ex:Thrain is a Dwarf, I have made a naïve consideration of ex:Thor also as Dwarf, for the shake of the first mapping (check below Table 1).

Based on the D defined above we have the following *pattern expressions* P.

$P_1 = (?N, rdf:type, ?TP)$

$P_2 = (?N, ex:hasFather, ?F)$

$P_3 = (?N, ex:hasTitle, ?TL)$

Then we define $\left[[P]\right]_D = ((P_1\ AND\ P_2)\ OPT\ P_3)$

Each *pattern expression* includes mappings $\mu_1 ... \mu_N$ such as $\mu \in \left[[P_i]\right]_D$

INPUT : An RDF dataset D, a *graph pattern* P and a mapping μ.

QUESTION : Is $\mu \in \left[[P_i]\right]_D$?

$\left[[P_1]\right]_D$=

	$?N$	$?TP$
μ_1:	ex:Thorin	ex:Dwarf
μ_2:	ex:Thrain	ex:Dwarf
μ_3:	ex:Thor	ex:Dwarf

Table 1

$\left[[P_2]\right]_D$=

	$?N$	$?F$
μ_1:	ex:Thorin	ex:Thrain
μ_2:	ex:Thrain	ex:Thor

Table 2

$\left[[P_1\ AND\ P_2]\right]_D = \left[[P_1]\right]_D \bowtie \left[[P_2]\right]_D$

	$?N$	$?TP$	$?F$
μ_1:	ex:Thorin	ex:Dwarf	ex:Thrain
μ_2:	ex:Thrain	ex:Dwarf	ex:Thor

Table 3

$\left[[P_3]\right]_D$=

	$?N$	$?TL$
μ_1:	ex:Thorin	"King under the Mountain"

Table 4

$\left[[P_1\ AND\ P_2]\ OPT\ P_3\right]_D = \left[\left[[P_1]\right]_D \bowtie \left[[P_2]\right]_D\right] \ \overline{\bowtie}\ \left[[P_3]\right]_D$

	$?N$	$?TP$	$?F$	$?TL$
μ_1:	ex:Thorin	ex:Dwarf	ex:Thrain	"King under the Mountain"
μ_2:	ex:Thrain	ex:Dwarf	ex:Thor	

Table 5

Question 2

(a) All postgrad students are students.

$PostgradStudent \sqsubseteq Student$

(b) Every postgrad student has bachelor degree.

$PostgradStudent \sqsubseteq \exists hasDegree.Bachelor$

(c) There exists one postgrad student.

$\exists x(PostgradStudent(x))$

(d) Every student follows some course.

$Student \sqsubseteq \exists follows.Course$

(e) Every student follows Knowledge Technologies course and some other course.

$Student \sqsubseteq \forall follows.KnowledgeTechnologies \sqcap \exists follows.Course$

(f) There exists one student that all other students dislike him.

$\exists x(Student(x) \wedge \forall y(Student(y) \wedge \neg(x = y) \Rightarrow Dislike(y, x)))$

(g) Peter is a person.

$Person(Peter)$

(h) Peter does not like the Knowledge Technologies course.

$\neg likes(Peter, KnowledgeTechnologies)$

(i) None of the students likes Peter.

$\forall x(Student(x) \Rightarrow \neg likes(x, Peter)$

or

$\neg\exists x(Student(x) \wedge likes(x, Peter))$

(j) Peter follows at least one course.

$\neg\exists x(Course(x) \wedge follows(x, Peter))$

(k) Paul does not follow any course.

$\forall x(Course(x) \Rightarrow \neg follows(Paul, x)$

(l) Every student follows at least one course.

$Student \sqsubseteq \exists follows.Course$

(m) Only one student failed the Knowledge Technologies course.

$\exists x \left(Student(x) \wedge failed(x, KnowledgeTechnologies) \right.$
$\left. \wedge \forall y \left(Student(y) \wedge failed(y, KnowledgeTechnologies) \Rightarrow x = y \right) \right)$

(n) None of the students failed the Knowledge Technologies course but at least one student failed at the Database course.

$\neg\exists x(Student(x) \wedge failed(x, KnowledgeTechnologies)) \wedge \exists x (Student(x) \wedge failed(x, Database))$

(o) Every student that follows the Knowledge Technologies course follows also the Logic Programming course.
$\forall x(Student(x) \wedge follows(x, KnowledgeTechnologies) \Rightarrow follows(x, Logic\ Programming))$

(p) There is no student that can fool all the other students.
$\neg \exists x(Student(x) \wedge \forall y\,(Student(y) \wedge \neg(x = y) \Rightarrow Fools(x, y)))$

(q) A biped is an animal that has exactly two legs.
$\text{Biped} \sqsubseteq Animal \sqcap (\geq 2 bodyPart.Legs) \sqcap (\leq 2 bodyPart.Legs)$

(r) A triangle is a polygon with exactly three edges and exactly three vertices which are line segments.
$Trianlge \equiv Polygon \sqcap (\geq 3 hasPart.Edges) \sqcap (\leq 3 hasPart.Edges) \sqcap (\geq 3 hasVertices.LineSegments) \sqcap (\leq 3 hasVertices.LineSegments)$

(s) A right-angled triangle is a triangle in which one angle is a right angle.
$RightAngledTrianlge \equiv Triangle \sqcap (\geq 1 angle.RightAngle) \sqcap (\leq 1 angle.RightAngle)$

Question 3

Sentence	**Evaluation**
$Person \sqcap hasChild$	Wrong
$\exists hasChild. \equiv Person$	Wrong
$\exists hasChild.(\geq 1)$	Wrong
$hasChild \subseteq hasBaby$	Correct
$hasChild(ANNA)$	Wrong
$Person \equiv \exists hasChild.\bot$	Correct

Question 4

Consider the following English sentences:

- Yannis is a person.
- The only kind of coffee that Yannis drinks is frappe.
- A Greek is a person who drinks only frappe coffee.
- Yannis is Greek.

a) Give an *ALC* knowledge base *KB* which formalizes the first three of the above sentences and an *ALC* formula φ that formalizes the fourth sentence.

A Knowledge base KB consists of two components: TBox and ABox. Thus, we must define those two concepts based on the first three sentences.

TBox : T

$T = \{$

$Greek \equiv Person \sqcap \forall\ drinks.Frappe$

}, I used ∀ (forAll, universal) symbol because in the range R of the things that someone can drink, Greek drink only Frappe coffee.

ABox : A

$A = \{$

$(Person \sqcap \forall drinks.Frappe)(YIANNIS)$

}

Formula φ = $Greek(YIANNIS)$

b) Construct an interpretation I which satisfies KB.

---

*Note: $Greek(YIANNIS)$ is not in the Knowledge Base so formula φ won't be included below.

N_c = { Person, Coffee, Frappe }

N_R = { drinks }

N_I = { *YIANNIS*}

Let Δ^I be the domain that will contain the following elements: Δ^I = { *Y, F, C, P* }

$\mathrm{I}_I(\mathrm{YIANNIS})^I = Y$

$\mathrm{I}_I(\mathrm{COFFEE})^I = C$, where C is a different type of coffee than F, $C \in N_i\ and\ C\ \approx F$

$\mathrm{I}_I(\mathrm{PERSON})^I = P$, where P is a different person than Y, , $P \in N_i\ and\ P\ \approx Y$

$\mathrm{I}_I(\mathrm{FRAPPE})^I = F$

$\mathrm{I}_c(Person)^I = \{ Y, P \}$

$\mathrm{I}_c(Coffee)^I = \{ C, F \}$

$\mathrm{I}_R(drinks)^I = \{ < Y, F >, < P, C > \}$

Give an Interpretation I that satisfies *KB*

$Coffee \sqsubseteq\ \neg \{ Y \}$ is valid, because $(\neg \{ Y)^I = \Delta^I \setminus (\{ Y \})^I = \Delta^I \setminus \{$ Y $\} = \{$ C, F $\}$ which is the set of $Coffee^I$ and Y is not included in that set.

c) Construct another interpretation I1 which does not satisfy KB.

Give an Interpretation I that is not satisfied by the *KB*.

$drinks(Y, C)$ is not valid since the pair of the respective individuals is not contained in the extension of the ***drinks*** role: $< YIANNIS^I, COFFEE^I > = < Y, C > \notin drinks^I$.

d) Use tableau techniques to prove that KB |= ϕ.

$KB = \{$

$$Greek \equiv Person \sqcap \forall drinks.Frappe,$$
$$Person \sqcap \forall drinks.Frappe(YIANNIS)$$

$\}$

First, we define a TBox that will contain the terminological axioms of the KB.

$T = \{ Greek \equiv Person \sqcap \forall drinks.Frappe \}$

We want to know if $\varphi = Greek(YIANNIS)$ is a logical consequence of the above defined KB. Thus, we add in the KB the axiom $Greek(YIANNIS)$. To show that φ is a logical consequence of KB, we must show that there is a model in T that satisfies the $\varphi = Greek(YIANNIS)$.

Let T be re-written in T' after unfolding the definition equivalence in T.

$T' = \{ Greek \subseteq (Person \sqcap \forall drinks.Frappe), (Person \sqcap \forall drinks.Frappe) \subseteq Greek \}$

$T' = \{ \neg Greek \sqcup (Person \sqcap \forall drinks.Frappe), \neg(Person \sqcap \forall drinks.Frappe) \sqcup Greek \}$

Then the Negation Normal Form of T' would be

$nnf(T') = \{ \neg Greek \sqcup (Person \sqcap \forall drinks.Frappe), \neg Person \sqcup \neg \forall drinks.Frappe \sqcup Greek \}$

$nnf(T') = \{ \neg Greek \sqcup (Person \sqcap \forall drinks.Frappe), \neg Person \sqcup \exists drinks.\neg Frappe \sqcup Greek \}$

1| $(Person \sqcap \forall drinks.Frappe)(YIANNIS)$ given

2| $Greek(YIANNIS)$ given

3| $\neg Greek \sqcup (Person \sqcap \forall drinks.Frappe)$ given from $nnf(T')$

4|$\neg Person \sqcup \exists drinks.\neg Frappe \sqcup Greek$ given from $nnf(T')$

5| $\neg Greek(YIANNIS)$ 3,$\sqcup -rule$

7| Clash 2,5

6| $(Person \sqcap \forall drinks.Frappe)(YIANNIS)$ 3,$\sqcup -rule$

8|$Person(YIANNIS)$ 7,$\sqcap -rule$

9|$(\forall drinks.Frappe)(YIANNIS)$ 7,$\sqcap -rule$

Branch 9 is already satisfied from the axiom in the defined KB.

10| $(Person \sqcap \forall drinks.Frappe)(YIANNIS)$ 1,$\sqcap -rule$

11| $Person(YIANNIS)$ 10,⊓ $-rule$

12|$(\forall drinks.Frappe)(YIANNIS)$ 7,⊓ $-rule$

Branch 12 is satisfied from branch 9. Concluding that YIANNIS drinks only Frappe.

13|$\neg Person(YIANNIS)$ 4,⊔ $-rule$ 14| $\exists drinks.\neg Frappe(YIANNIS)$ 4,⊔ $-rule$ 15|$Greek(YIANNIS)$ 4,⊔ $-rule$

16| Clash 1,14 17| $drinks(YIANNIS, a)$ 8,∃ $-rule$

18| $\neg Frappe(a)$ 8,∃ $-rule$

19| $Frappe(a)$ 9, ∀ $-rule$

20| Clash 13,14

We have found at least one branch, specifically branch 15, where no clash/contradiction exists. Also, no rules can be applied to further expand the branch. Thus, we have found a model that satisfies the assumption KB |= *φ*.

Question 5

- Yannis is a person.
- Frappe is a kind of coffee. Nescafe frappe is a kind of frappe.
- The only kind of coffee that Yannis drinks is Nescafe frappe.
- A Greek is a person who drinks only Frappe coffee.
- Yannis is Greek.

TBox : T

$T = \{$

$Greek \equiv Person \sqcap \forall drinks.Frappe,$

$Frappe \subseteq Coffee,$

$Frappe(NESCAFE)$

}, I used ∀ (forAll, universal) symbol because in the range R of the things that someone can drink, Greek drink only Frappe coffee.

ABox : A

$A = \{ Person(YIANNIS), drinks(YIANNIS, NESCAFE) \}$

$KB = \{$

$Greek \equiv Person \sqcap \forall drinks.Frappe,$

$Frappe(NESCAFE),$

$Frappe \subseteq Coffee,$

$Person(YIANNIS),$

$drinks(YIANNIS, NESCAFE)$

}

Formula $\varphi = Greek(YIANNIS)$

Given the above knowledge base KB = (T, A) prove that it exists a formula φ such that KB|=φ with $\varphi = Greek(YIANNIS)$.

We want to know if $\varphi = Greek(YIANNIS)$ is a logical consequence of the above defined KB and φ can be found in a model that is satisfied by the defined KB. Thus, we add in the KB the axiom $Greek(YIANNIS)$. To show that φ is a logical consequence of KB, we must show that there is a model in KB that satisfies the $\varphi = Greek(YIANNIS)$.

Let T be re-written in T' after unfolding the definitions and subsumptions of T.

$T' = \{$

$$Greek \sqsubseteq (Person \sqcap \forall drinks.Frappe),$$
$$(Person \sqcap \forall drinks.Frappe) \sqsubseteq Greek,$$
$$Frappe \sqsubseteq Coffee$$

$\}$

$T' = \{$

$$\neg Greek \sqcup (Person \sqcap \forall drinks.Frappe),$$
$$\neg(Person \sqcap \forall drinks.Frappe) \sqcup Greek,$$
$$\neg Frappe \sqcup Coffee$$

$\}$

Then the Negation Normal Form of T' would be

$nnf(T') = \{$

$$\neg Greek \sqcup (Person \sqcap \forall drinks.Frappe),$$
$$\neg Person \sqcup \neg\forall drinks.Frappe \sqcup Greek,$$
$$\neg Frappe \sqcup Coffee$$

$\}$

$nnf(T') = \{$

$$\neg Greek \sqcup (Person \sqcap \forall drinks.Frappe),$$
$$\neg Person \sqcup \exists drinks.\neg Frappe \sqcup Greek,$$
$$\neg Frappe \sqcup Coffee$$

$\}$

The knowledge base can now be defined as $KB = \{nnf(T'), A\}$, with $YIANNIS$ and $NESCAFE$ instances of the KB.

1| $Person(YIANNIS)$ given

2| $Frappe(NESCAFE)$ given

3| $drinks(YIANNIS, NESCAFE)$ given

4| $Greek(YIANNIS)$ given

5| $\neg Greek \sqcup (Person \sqcap \forall drinks.Frappe)(YIANNIS)$ given from $nnf(T')$

6| $\neg Person \sqcup \exists drinks.\neg Frappe \sqcup Greek(YIANNIS)$ given from $nnf(T')$

7| $\neg Frappe \sqcup Coffee(NESCAFE)$ given from $nnf(T')$

8| $\neg Greek(YIANNIS)$ 5,$\sqcup -rule$

9| $(Person \sqcap \forall drinks.Frappe)(YIANNIS)$ 5,$\sqcup -rule$

10| **Clash 4,8**

11| $Person(YIANNIS)$ 9,$\sqcap -rule$

12| $(\forall drinks.Frappe)(YIANNIS)$ 9,$\sqcap -rule$

13| $Frappe(NESCAFE)$ 12,$\forall -rule$

14|$\neg Person(YIANNIS)$ 6,$\sqcup -rule$

15| $\exists drinks.\neg Frappe(YIANNIS)$ 6,$\sqcup -rule$

16|$Greek(YIANNIS)$ 6,$\sqcup -rule$

17| **Clash 1,14**

18| $drinks(YIANNIS, NESCAFE)$ $15, \exists -rule$

19| $\neg Frappe(NESCAFE)$ $15, \exists -rule$

20| **Clash 2,19**

21| $\neg Frappe(NESCAFE)$ 7,$\sqcup -rule$

22| $Coffee(NESCAFE)$ 7,$\sqcup -rule$

Clash 2,21

The subsumption algorithm determines subconcept-superconcept relationships: C is subsumed by D iff all instances of C are necessarily instances of D, i.e., the first concept is always interpreted as a subset of the second concept. (Baader)

Since Nescafe is a kind of Frappe then from the subsumption $Frappe \subseteq Coffee$ Nescafe is also of kind Coffee.

Thus, in branch 22 we have no clash/contradiction, and no further rules can be applied to expand this branch.

We have found at least one branch, specifically branches 13, 16, 22, where no clash exists. We mostly care for branch 16 where the formula φ is satisfied. However, I have also unfolded all the Tboxes. Thus, we have found a model that satisfies the assumption KB |= φ.

Question 6

Consider the following English sentences:

- Walt is a person.
- Walt has three distinct pets: Huey, Dewey and Louie.
- Huey, Dewey and Louie are animals.
- An animal lover is a person which has at least three pets that are animals.
- Walt is an animal lover.

TBox : T

$T = \{$

$AnimalLover \equiv Person \sqcap (\geq 3 hasPet.Animal)$

}

ABox : A

Assertions can be used to state facts about named individuals.

$A = \{$

$Person(WALT)$, -> Concept assertion among named Individuals

$Animal(HUEY)$,

$Animal(DEWEY)$,

$Animal(LOUIE)$,

$hasPet(WALT, HUEY)$, -> Role assertion among named Individuals

$hasPet(WALT, DEWEY)$,

$hasPet(WALT, LOUIE)$

}

Formula $\varphi = AnimalLover(WALT)$

Given the above knowledge base KB = (T, A) prove that it exists a formula φ such that KB|=φ with $\varphi = AnimalLover(WALT)$.

We want to know if φ = $AnimalLover(WALT)$ is a logical consequence of the above defined KB and φ can be found in a model that is satisfied by the defined KB. Thus, we add in the KB the axiom $AnimalLover(WALT)$.
To show that φ is a logical consequence of KB, we must show that there is a model in KB that satisfies the φ = $AnimalLover(WALT)$.

$KB = \{$

$$AnimalLover \equiv Person \sqcap (\geq 3 hasPet.Animal),$$
$$Person(WALT),$$
$$Animal(HUEY),$$
$$Animal(DEWEY),$$
$$Animal(LOUIE),$$
$$hasPet(YIANNIS, HUEY),$$
$$hasPet(YIANNIS, DEWEY),$$
$$hasPet(YIANNIS, LOUIE),$$
$$AnimalLover(WALT)$$

}

Let T be re-written in T' after unfolding the definitions and subsumptions of T.

$T' = \{$

$AnimalLover \subseteq (Person \sqcap (\geq 3hasPet.Animal)), (Person \sqcap (\geq 3hasPet.Animal)) \subseteq AnimalLover$

}

$T' = \{$

$\neg AnimalLover \sqcup (Person \sqcap (\geq 3hasPet.Animal)), \neg(Person \sqcap (\geq 3hasPet.Animal)) \sqcup AnimalLover$

}

Then the Negation Normal Form of T' would be

$nnf(T') = \{$

$\neg AnimalLover \sqcup (Person \sqcap (\geq 3hasPet.Animal)),$

$\neg Person \sqcup \neg(\geq 3hasPet.Animal) \sqcup AnimalLover$

}

$nnf(T') = \{$

$\neg AnimalLover \sqcup (Person \sqcap (\geq 3hasPet.Animal)),$

$\neg Person \sqcup (\leq 2hasPet.Animal) \sqcup AnimalLover$

}

The knowledge base can now be defined as $KB = \{ nnf(T'), A\}$, with $WALT$ as instance of the KB. Next, we apply the tableau rules on this KB.

1| $AnimalLover(WALT)$ given

2| $Person(WALT)$ given

3| $hasPet(WALT, HUEY)$ given

4| $hasPet(WALT, DEWEY)$ given

5| $hasPet(WALT, LOUIE)$ given

6| $Animal(HUEY)$ given

7| $Animal(DEWEY)$ given

8| $Animal(LOUIE)$ given

9| $(\neg Person \sqcup (\leq 2hasPet.Animal) \sqcup AnimalLover)(WALT)$ given from $nnf(T')$

10| $(\neg AnimalLover \sqcup (Person \sqcap (\geq 3hasPet.Animal)))(WALT)$ given from $nnf(T')$

11| $\neg Person(WALT)$ 9,$\sqcup -rule$

12| $\leq 2hasPet.Animal(WALT)$ 9,$\sqcup -rule$

13| $AnimalLover(WALT)$ 9,$\sqcup -rule$

14| **Clash 2,10**

15| $hasPet(WALT, y_1)$, where $\mathcal{L}(y_1) = HUEY = \{Animal\}$

16| $hasPet(WALT, y_2)$, where $\mathcal{L}(y_2) = DEWEY = \{Animal\}$, $y_1 \not\approx y_2$

17| **Clash because individual $WALT$ has also a third pet $LOUIE$**. Contradiction from branches 3,4,5

18| $\neg AnimalLover(WALT)$ 10,$\sqcup -rule$

19| $(Person \sqcap (\geq 3hasPet.Animal))(WALT)$ 10,$\sqcup -rule$

20| **Clash 1,18**

21| $Person(WALT)$ 19,$\sqcap -rule$

22| $(\geq 3hasPet.Animal)(WALT)$ 19,$\sqcap -rule$

23| $hasPet(WALT, y_1)$, where $\mathcal{L}(y_1) = HUEY = \{Animal\}$,

24| $hasPet(WALT, y_2)$, where $\mathcal{L}(y_2) = DEWEY = \{Animal\}$,

25| $hasPet(WALT, y_3)$, where $\mathcal{L}(y_3) = LOUIE = \{Animal\}$, $and\ y_1 \neq y_2 \neq y_3$

No contradiction because $WALT$ has indeed three pets y_1, y_2, y_3 based on the given KB and branches 3,4,5

Branches 13, 25 have no contradiction with the given Knowledge Base KB. And no further rule can be applied on those two branches. Thus, we have proved that $AnimalLover(WALT)$ can be a logical consequence of the define Knowledge Base KB.

Question 7

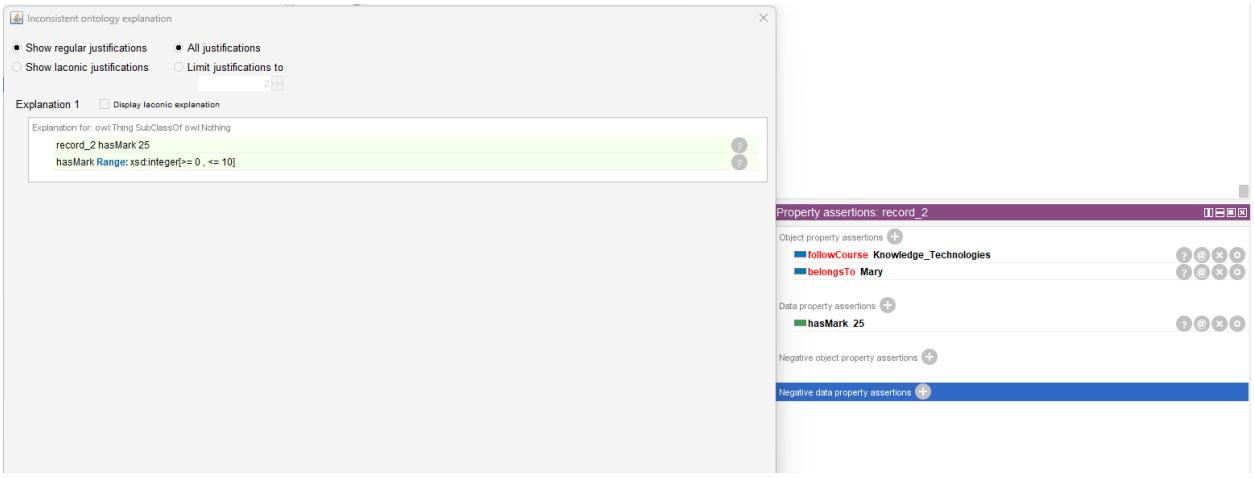
(a) You may find the University ontology (.owl file) created in Protégé-5.5.0 in the project deliverables folder by the name "university_ontology_v3.owl".

Question: What can you infer for Maria and Laura?

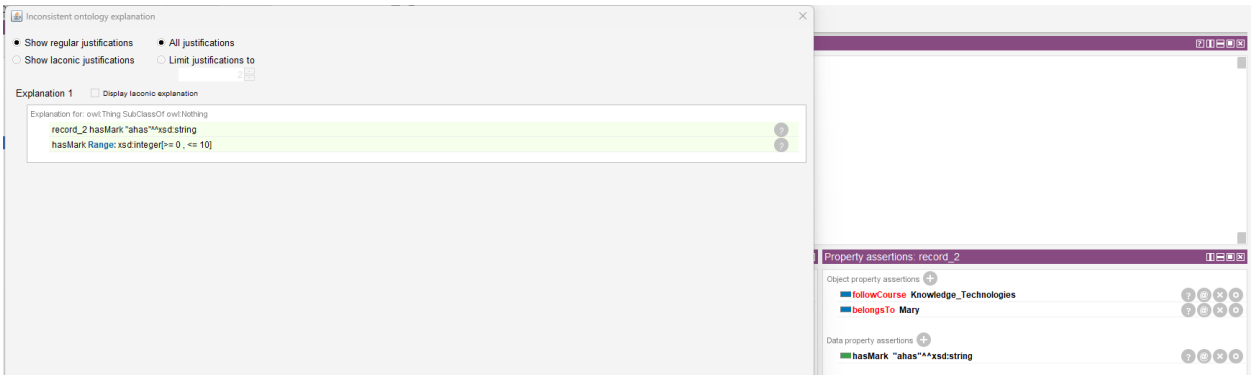
Answer: Based on the created ontology we can infer for Maria (aka Mary in the ontology delivered) that she is a Postgraduate student because she follows at least one Postgraduate Course. Laura wasn't classified as a Postgraduate student or an Undergraduate student. She wasn't inferred as a student of any of the two University programs. This was an expected outcome since if a student follows one Undergraduate course, s/he can be registered in either a Postgraduate or an Undergraduate program.

Below I have some examples to test some of the constraints of the ontology.

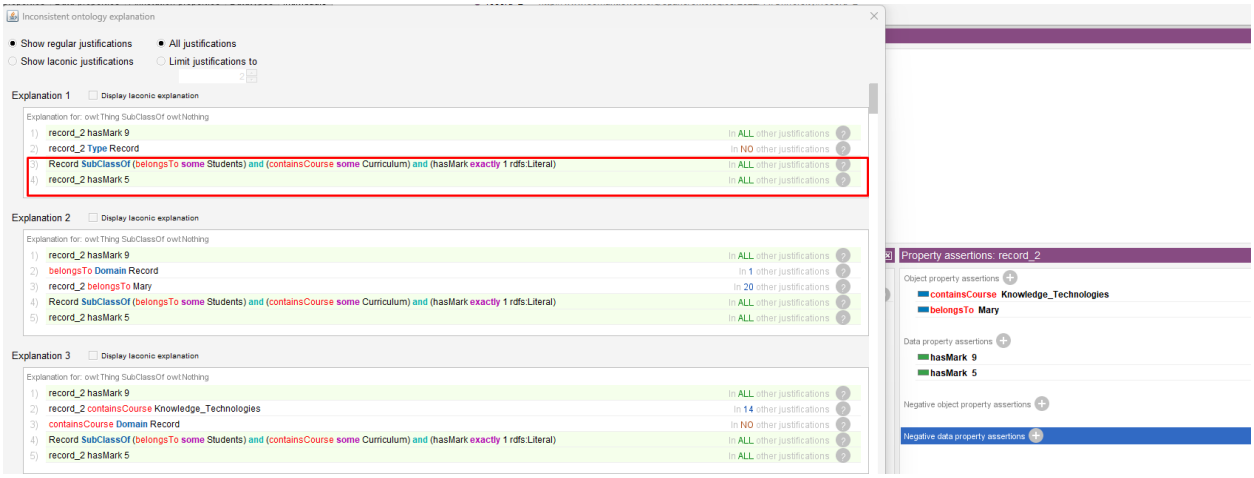
Constraint 1: Insert mark for a course out of the specified range (>10 or <0).



Constraint 2: Insert mark for a course as string value.

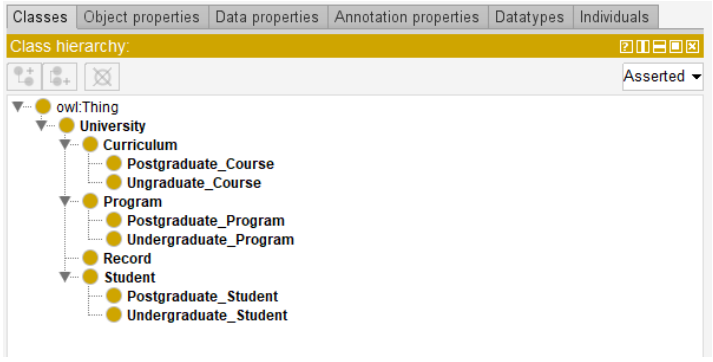


Constraint 3: Mark a course twice in the same record.



(b) Testing the ontology with Cha2O.

After using the tool Cha2O I have created the same ontology structure as in the first part of the exercise (a). The file of the ontology is named “try_v3.owl” and can be found in the project’s deliverables. The file when inserted in Protégé will generate the following structure for the University ontology,



The same exact structure as in the first part (a) of the exercise.

Question: (a) How much of the specific fragment of the ontology did you manage to recreate? (1 for none of it, 10 for all of it)

Grade: 6/10.

Explanation: I managed to create the hierarchy classes of the ontology. The four major sub-classes of university class (Curriculum-Courses, Program, Student, Record) along with their sub-classes (i.e., Undergraduate_Student, Postgraduate_Courses, etc). I have also created the object properties defining relationships between the concepts of the ontology. One major difference with the Protégé tool is the name given to the relationships. For example, in Protégé I had "followCourse" with Domain *Student* and Range *Curriculum.* With the tool this relationship is named as *Student_Follows_Curriculum.* The annotations could be also generated/imported with the tool. The data properties, individuals, and qualified number restrictions (such as exactly, up to two) could not be developed with the Cha2O tool.

Question: (b) How helpful was the tool?

Grade: 7/10.

Explanation: The three points were deducted for the installation process. Starting from there it was quite tedious to follow all the steps with specific Python installation and extra packages. A better way could be the creation of a docker image with all the libraries and python binaries pre-installed. After launching the tool, the rest of the steps were well-explained. It was very helpful for the functionality intended. It was also very helpful the way to type Natural Language sentences and create the Object properties and relationships from that. I have tested the ontology result by importing the saved .owl file to Protégé and I have verified that Domain and Range was successfully created/inferred from a simple English sentence.

Question: (c) Would you reuse such a tool for future projects with ontologies?

Grade: 8/10.

Explanation: Two points deducted for the lack of functionalities from other open-source tools. Moreover, I would be more willing to use after some fixes on the general user experience that I will explain below in the coming questions. Overall, the tool it does what is designed for.

Question: (d) What level of KT-knowledge do you think is required to use this tool? (1 for none of it, 10 for very good knowledge of KTs).

Grade: 5/10.

Explanation: Only a very basic knowledge on terminologies like Super-classes, Sub-classes, and Relationships between Concepts. The user should be aware of the actions s/he takes within the tool. For example, when a user creates a

Specialization for a Super-class then it is common-knowledge that every Sub-class is also of type Super-class (i.e. Every Postgraduate_Student is also of type Student).

Question: (e) How familiar were you with the language used?

Grade: 3/10.

Explanation: I wasn't familiar with the language used. Knowing only the basic concepts of Knowledge Graphs and OWL ontologies and with the help of the explanation boxes of the tool it was easy for me to quickly understand the tool's functionality.

Question: (f) Do the explanations given by the tool suffice to complete your task?

Grade: 5/10.

Explanation: The tabs of *Generalize* and *Specialize* were clear enough. However, in the description of the tab *Competency Question* it is described that a user can define Class hierarchies. It's not clear how the user can achieve this also given the illustrated examples in the tool. I have achieved the class hierarchy by following the instructions in the *Specialize* tab. But I had to follow some trial & error runs to achieve the desired result (i.e., Undergraduate_Course isSubclassOf Course).

Question: (g) To what level did the tool require from you some extra effort (e.g., to memorize information)

Grade: 8/10.

Explanation: I had to create different versions of the same ontology to keep track of the changes since the revert of a change/mistake was not possible. Also, I kept screenshots of the launching page and of additional information if needed because most of the explanations/examples were shown only after clicking a specific button.

Question: (h) To what level were you aware of what the tool was doing at each step?

Grade: 8/10.

Explanation: The tool is full of pop-up messages for when the Domain, Range of Object properties are created, or when the ontology is saved. This behavior was very friendly to the user. Two points deducted because it wasn't clear to me of what the tool will generate after specifying comma-separated sub-classes to specialize a super-class. I had to test this behavior to find out the expected result.

Question: (i) Did you often face error messages (how did you deal with them?)?

Grade: 6/10.

Explanation: Some of the errors/bugs encountered:

1) The application was exiting with error message if no file was selected after clicking the *Find File* button, which would locate the file locally. Thus, the application should be started again.
2) You can't undo mistakes, with the application forcing you to keep different versions of the ontology.
3) If you apply an object property between a *Domain* and a *Range,* then a pop-up message is generated denoting the relationship defined and prompts the user's acceptance for *Yes/No*. Clicking *No* the object property is still created. This is a major bug of the tool, and it was observed on more than two similar pop-up windows (Confirmation window in *Specialize* button and in *Competency Question* button).

While using the tool I didn't manage to recreate the Data Properties, the Qualified Number Restrictions and the specific Individuals that were part of the ontology. The individuals such as the records, or the students were not re-created. It would be helpful to add an undo mechanism. I also didn't manage to annotate the specific restrictions between different concepts (keywords like *exactly, single, some, only).* Additionally, as noted earlier, make the installation process more user-friendly by packaging the tool into a docker image.

References

Baader, F. (n.d.). *Description Logics.*