Assignment 5

Course Lecturer: Yizheng Zhao

June 1, 2023

⋆ **This assignment, which is due on June 30th, contributes to 10% of the final marks for this course. Moreover, the last four questions present an opportunity for students to secure up to four additional bonus marks. These bonus marks can potentially increase a student's overall marks but are subject to a maximum total of 100 for the course.**

Question 1. CWA and OWA, querying without TBox
Consider the database instance $\mathcal{D}_{\text{music}}$ given by:

StudioAlbum(Fantasy) StudioAlbum(The_Eight_Dimensions) StudioAlbum(Common_Jasmin_Orange)

DebutAlbum(Jay) LiveAlbum(2004_Incomparable_Concert)

EP(Hidden_Track) EP(Initial_D) SoundtrackAlbum(Secret) CompilationAlbum(Together)

Song(Herbalist_Manual) Song(Elimination) Singer(Jay_Chou) Singer(Eason_Chan)

Composer(Jay_Chou) Lyricist(Vincent_Fang) Police(Black_Cat)

hasFriend(Jay_Chou, Vincent_Fang) hasFriend(Jay_Chou, Will_Liu)

releases(Jay_Chou, Jay) releases(Jay_Chou, Fantasy) releases(Jolin_Tsai, Together)

sings(Eason_Chan, Elimination) sings(Jolin_Tsai, Rewind) sings(Jay_Chou, Herbalist_Manual)

writesMusicFor(Jay_Chou, Elimination) writesMusicFor(Jay_Chou, Rewind)

writesMusicFor(Jay_Chou, Herbalist_Manual) writesMusicFor(Ta-yu_Lo, Pearl_of_the_Orient)

writesLyricsFor(Jay_Chou, Elimination) writesLyricsFor(Vincent_Fang, Rewind)

writesLyricsFor(Vincent_Fang, Herbalist_Manual) DancesWith(Will_Liu, Herbalist_Manual)

Consider each of the following Boolean queries F (in DL notation).

1. Album(Fantasy)
2. StudioAlbum(The_Eight_Dimensions)
3. LiveAlbum(Common_Jasmin_Orange)
4. ¬LiveAlbum(Common_Jasmin_Orange)
5. ¬EP(secret)
6. ¬StudioAlbum ⊔ ¬LiveAlbum(2004_Incomparable_Concert)

7. $\neg$StudioAlbum $\sqcup$ $\neg$LiveAlbum(Eason_Chan)
8. $\exists$hasFriend.$\top$(Jay_Chou)
9. $\exists$hasFriend.$\exists$dancesWith.Song(Jay_Chou)
10. $\exists$hasFriend.Composer(Jay_Chou)
11. $\exists$hasFriend.{Jay_Chou}(Vincent_Fang)
12. DebutAlbum(2004_Incomparable_Concert)
13. Song(Rewind)
14. Singer(Jay_Chou)
15. Singer(Jolin_Tsai)
16. Lyricist(Jay_Chou)
17. Composer(Jay_Chou)
18. Composer(Ta-yu_Lo)
19. Police(Jay_Chou)
20. Police(Jolin_Tsai)
21. $\neg$Singer-SongWriter $\sqcup$ $\neg$Police(Vincent_Fang)
22. $\neg$Singer-SongWriter $\sqcup$ $\neg$Police(Ta-yu_Lo)
23. Singer-SongWriter(Jay_Chou)
24. Singer-SongWriter(Jolin_Tsai)
25. $\neg$SongWriter(Vincent_Fang)
26. $\neg$Dancer(Will_Liu)

- Write those Boolean queries marked in red in first-order logic (FOL) notation. (Note that for many queries there is no difference between DL notation and FOL notation).
- Query answering under CWA: check for each Boolean F whether the answer to the query F given by $\mathcal{D}_{\text{music}}$ is "Yes" or "No".
- Query answering under OWA: check for each Boolean query F whether the certain answer to F given by $\mathcal{D}_{\text{music}}$ is "Yes", "No", or "Don't know".

Consider the following non-Boolean queries F_i ($1 \leq i \leq 4$):

- $F_1(x) = \text{Singer}(x)$
- $F_2(x) = \neg\text{Singer}(x)$
- $F_3(x, y) = \text{hasFriend}(x, y)$
- $F_4(x) = (\text{Lyricist}(x) \vee \text{Composer}(x)) \wedge \neg\text{releases}(x, \text{Jay})$

For each query F_i, give

- for CWA: answer(F_i,$\mathcal{D}_{\text{music}}$);
- for OWA: certanswer(F_i,$\mathcal{D}_{\text{music}}$).

Question 2. Querying with TBox

Following Question 1, consider now the TBox $\mathcal{T}$ given as:

$$\begin{aligned}
\mathsf{StudioAlbum} &\sqsubseteq \mathsf{Album} \\
\mathsf{LiveAlbum} &\sqsubseteq \mathsf{Album} \\
\mathsf{StudioAlbum} \sqcap \mathsf{LiveAlbum} &\sqsubseteq \bot \\
\mathsf{EP} \sqcap \mathsf{LiveAlbum} \sqcap \mathsf{SoundTrackAlbum} &\sqsubseteq \bot \\
\mathsf{DebutAlbum} &\sqsubseteq \mathsf{StudioAlbum} \\
\mathsf{Album} &\equiv \exists \mathsf{hasTrack}.\mathsf{Song} \\
\mathsf{Singer} &\equiv \exists \mathsf{releases}.\mathsf{Album} \sqcap \exists \mathsf{sings}.\mathsf{Song} \\
\exists \mathsf{releases}^{-}.\mathsf{Police} \sqcap \mathsf{Album} &\sqsubseteq \bot \\
\mathsf{Lyricist} &\equiv \exists \mathsf{writesLyricsFor}.\mathsf{Song} \\
\mathsf{Composer} &\sqsubseteq \exists \mathsf{writesMusicFor}.\mathsf{Song} \\
\mathsf{SongWriter} &\sqsubseteq \mathsf{Lyricist} \sqcap \mathsf{Composer} \\
\mathsf{Singer\text{-}SongWriter} &\sqsubseteq \mathsf{Singer} \sqcap \mathsf{SongWriter} \\
\exists \mathsf{sings}^{-}.\top &\sqsubseteq \mathsf{Song} \\
\exists \mathsf{writesLyricsFor}^{-}.\top &\sqsubseteq \mathsf{Song}
\end{aligned}$$

- Reconsider the Boolean queries F given in Question 1. Compute the certain answers to each F in the context of $(\mathcal{T}, \mathcal{D}_{\text{music}})$.
- The inclusion of the TBox $\mathcal{T}$ to $\mathcal{D}_{\text{music}}$ allows one to draw new conclusions from $\mathcal{D}_{\text{music}}$ and may render some of the data (ABox assertion) α in $\mathcal{D}_{\text{music}}$ redundant, i.e., $(\mathcal{T}, \mathcal{D}_{\text{music}} \backslash \{\alpha\}) \models \mathcal{D}_{\text{music}}$. As such, can you identify all redundant assertions α?

Question 3. Computing $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ in $\mathcal{EL}$

Consider the following $\mathcal{EL}$ TBox $\mathcal{T}$:

$$\begin{aligned}
\mathsf{Vocalist} &\sqsubseteq \exists \mathsf{plays_for}.\mathsf{Band} \\
\mathsf{Guitarist} &\sqsubseteq \exists \mathsf{plays_for}.\mathsf{Band} \\
\mathsf{Bassist} &\sqsubseteq \exists \mathsf{plays_for}.\mathsf{Band} \\
\mathsf{Drummer} &\sqsubseteq \exists \mathsf{plays_for}.\mathsf{Band} \\
\mathsf{LeadGuitarist} &\sqsubseteq \mathsf{Guitarist} \\
\mathsf{RhythmGuitarist} &\sqsubseteq \mathsf{Guitarist} \\
\mathsf{Band} &\sqsubseteq \exists \mathsf{captained_by}.\mathsf{Captain} \\
\mathsf{Band} &\sqsubseteq \exists \mathsf{managed_by}.\mathsf{Manager} \\
\mathsf{Manager} &\sqsubseteq \exists \mathsf{managed_by}.\mathsf{Manager}
\end{aligned}$$

and the following ABox $\mathcal{A}$:

Captain(Monster)	Vocalist(Ashin)
LeadGuitarist(Monster)	Bassist(Masa)
RhythmGuitarist(Stone)	Drummer(Ming)
Band(Mayday)	managed_by(Mayday, Amuse)

- Compute the interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ as described in the slides.
- For $\mathcal{EL}$ concept queries, we know that $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ gives the answer "Yes" iff $(\mathcal{T},\mathcal{A})$ gives the certain answer "Yes". Check this for the following queries:
  - $\exists$plays_for.Band(Ashin);
  - $\exists$managed_by.Manager(Masa);
  - $\exists$plays_for.$\exists$captained_by.Captain(Monster);
  - $\exists$plays_for.$\exists$managed_by.Manager(Ming).
- For more complex queries, $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ can give the answer "Yes" even if $(\mathcal{T},\mathcal{A})$ does not give the certain answer "Yes". Check this for:
  - $F(x,y) = \exists z.(\text{plays\_for}(x,z) \wedge \text{plays\_for}(y,z))$.
  - $F = \exists x.\text{managed\_by}(x,x)$.

Question 4. SQL and Conjunctive Queries

Consider the following database $\mathcal{D}$ consisting of the following tables:

Person:	
ID	Name
2101	Jay_Chou
2102	Jolin_Tsai
2103	Stefanie_Sun
2104	Ta-yu_Lo

Enrollment:	
StudentID	Since
2102	2020
2103	2021
2104	2020

Attendance:	
StudentID	CourseID
2101	30000160
2102	30000160
2102	30000170
2103	30000150

Course:	
ID	Title
30000150	ML
30000160	KRP
30000170	NLP

- Define the finite first-order interpretation $\mathcal{I}_{\mathcal{D}}$ corresponding to $\mathcal{D}$.
- Reformulate each of the following SQL queries Q into first-order queries f_Q, and identify which of them are conjunctive queries.

- Answer Q in the context of $\mathcal{D}$ and f_Q in the context of $\mathcal{I}_\mathcal{D}$.

1. SELECT $*$ FROM Person
2. SELECT Person.Name FROM Person, Attendance, Course
   WHERE Person.ID = Attendance.PersonID
   AND Course.ID = Attendance.CourseID
   AND Course.Title = "KRP"
3. SELECT Person.Name FROM Person, Enrollment
   WHERE Person.ID = Enrollment.PersonID
   AND NOT EXISTS (
   SELECT $*$ FROM Attendance
   WHERE Person.ID = Attendance.PersonID)

Question 5. Certain Answers in Different Contexts

Consider the following $\mathcal{ALC}$ knowledge base $\mathcal{K} := (\mathcal{T}, \mathcal{A})$ with:

$$\mathcal{T} := \{\mathsf{X} \sqsubseteq Y, Y \sqsubseteq \exists r.X, X \sqsubseteq \forall r.Y, \forall r.X \sqsubseteq Y, W \equiv \neg V, \exists r.Y \sqsubseteq \neg V\}$$
$$\mathcal{A} := \{(\text{Jay\_Chou}, \text{Jolin\_Tsai}) : r, (\text{Jolin\_Tsai}, \text{Stefanie\_Sun}) : r, (\text{Stefanie\_Sun}, \text{Jay\_Chou}) : r,$$
$$(\text{Jolin\_Tsai}, \text{Jolin\_Tsai}) : r, (\text{Stefanie\_Sun}, \text{Stefanie\_Sun}) : r, \text{Stefanie\_Sun} : X\}$$

- Compute the certain answers to the following conjunctive queries in the context of $\mathcal{A}$.
- Compute the certain answers to the following conjunctive queries in the context of $\mathcal{K}$.

1. $r(x, y) \wedge Y(y)$
2. $\exists y(r(x, y) \wedge Y(y))$
3. $\exists x, y(r(x, y) \wedge r(y, x))$
4. $\exists z, w(r(x, y) \wedge r(y, z) \wedge r(z, x) \wedge r(z, w) \wedge W(w))$

Question 6. Reasoning in $\mathcal{EL}$

Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox containing:

$$\mathsf{X} \sqsubseteq \exists \mathsf{r.Y}, \mathsf{Y} \sqsubseteq \exists \mathsf{r.Y},$$

and $\mathcal{A}$ be an $\mathcal{EL}$-ABox containing:

$$\mathsf{r(a,b)}, \mathsf{X(b)},$$

- Compute the interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ so that all $\mathcal{EL}$-concepts C and d $\in \{a, b\}$:

$$\mathcal{T}, \mathcal{A} \models \mathsf{C(d)} \iff \mathcal{I}_{\mathcal{T},\mathcal{A}} \models \mathsf{C(d)}$$

Question 7. Ontology-Mediated Query Answering

Consider the following $\mathcal{ALC}$-TBox $\mathcal{T}$:

$$\top \sqsubseteq \mathsf{red} \sqcup \mathsf{green}, \mathsf{red} \sqcap \exists \mathsf{r.green} \sqsubseteq \mathsf{clash}, \mathsf{green} \sqcap \exists \mathsf{r.red} \sqsubseteq \mathsf{clash}$$

and the following $\mathcal{ALC}$-ABox $\mathcal{A}$:

$$\mathsf{r(0,\ 1)}, \mathsf{r(1,\ 2)}, \mathsf{r(2,\ 3)}, \mathsf{r(3,\ 0)}.$$

- What is the answer of $(\mathcal{T}, \mathcal{A})$ to the Boolean query $\exists x\ \mathsf{clash}(x)$? Explain your answer.

Question 8. Conservative Extension-based Module Extraction

The design, development, maintenance, reuse, and integration of ontologies are complex tasks. Like software engineers, ontology engineers need to be supported by tools and methodologies that help them to minimize the introduction of errors, i.e., to ensure that ontologies are consistent and do not have unexpected logical consequences. In order to develop this support, important notions from the field of software engineering, such as *module*, *black-box behavior*, and *controlled interaction*, need to be adapted.

For example, consider a scenario where an ontology engineer is building an ontology about research projects, which specifies different types of projects according to the research topic they focus on. While the ontology engineer is an expert on research projects—(s)he knows, for example, that an EUProject is a Project (axiom P3)—(s)he may not be well-versed in the specific topics covered by the projects, particularly when it comes to terms such as "Cystic_Fibrosis" and "Genetic_Disorder" mentioned in axioms P1 and P2. To address this knowledge gap, the engineer decides to leverage the knowledge available in a well-established and widely-used medical ontology $\mathcal{Q}$ containing the axioms M1—M5 in Figure 1, where the concepts Cystic_Fibrosisis and Genetic_Disorder are described.

	Ontology of medical research projects $\mathcal{P}$:
P1	Genetic_Disorder_Project $\equiv$ Project $\sqcap \exists$has_Focus.Genetic_Disorder
P2	Cystic_Fibrosis_EUProject $\equiv$ EUProject $\sqcap \exists$has_Focus.Cystic_Fibrosis
P3	EUProject $\sqsubseteq$ Project
P4	$\exists$has_Focus.$\top \sqsubseteq$ Project
E1	Project $\sqcap$ (Genetic_Disorder $\sqcap$ Cystic_Fibrosis) $\sqsubseteq \bot$
E2	$\forall$has_Focus.Cystic_Fibrosis $\sqsubseteq \exists$has_Focus.Genetic_Disorder
	Ontology of medical terms $\mathcal{Q}$:
M1	Cystic_Fibrosis $\equiv$ Fibrosis $\sqcap \exists$located_In.Pancreas $\sqcap \exists$has_Origin.Genetic_Origin
M2	Genetic_Fibrosis $\equiv$ Fibrosis $\sqcap \exists$has_Origin.Genetic_Origin
M3	Fibrosis $\sqcap \exists$located_In.Pancreas $\sqsubseteq$ Genetic_Fibrosis
M4	Genetic_Fibrosis $\sqsubseteq$ Genetic_Disorder
M5	DEFBI_Gene $\sqsubseteq$ Immuno_Protein_Gene $\sqcap \exists$associated_With.Cystic_Fibrosis

Figure 1: Reusing medical terminology in an ontology on research projects

The most straightforward way to reuse these concepts is to import into $\mathcal{P}$ the ontology $\mathcal{Q}$—that is, to add the axioms from $\mathcal{Q}$ into $\mathcal{P}$ and work with the extended ontology $\mathcal{P} \cup \mathcal{Q}$. Importing additional axioms into an ontology may result in new logical consequences. For example, it is easy to see that axioms M1—M4 in $\mathcal{Q}$ imply that every instance of Cystic_Fibrosis is an instance of Genetic_Disorder:

$$\mathcal{Q} \models \alpha := \text{Cystic\_Fibrosis} \sqsubseteq \text{Genetic\_Disorder} \tag{1}$$

Indeed, the axiom

$$\alpha_1 := \text{Cystic\_Fibrosis} \sqsubseteq \text{Genetic\_Fibrosis} \tag{2}$$

follows from the axioms M1 and M2 as well as from the axioms M1 and M3; α follows from the axioms α_1 and M4. Using α as well as the axioms P1—P3 in the ontology $\mathcal{P}$ we can now prove that every instance of Cystic_Fibrosis_EUProject is also an instance of Genetic_Disorder_Project:

$$\mathcal{P} \cup \mathcal{Q} \models \beta := \text{Cystic\_Fibrosis\_EUProject} \sqsubseteq \text{Genetic\_Disorder\_Project} \tag{3}$$

This axiom β, however, does not follow from $\mathcal{P}$ alone:

$$\mathcal{P} \not\models \beta := \text{Cystic\_Fibrosis\_EUProject} \sqsubseteq \text{Genetic\_Disorder\_Project} \tag{4}$$

The ontology engineer might not be aware of Entailment (3), even though it concerns the terms of primary scope in his/her projects ontology $\mathcal{P}$.

It could be expected that entailments like α in (1) from an imported ontology $\mathcal{Q}$ result in new logical consequences, like β in (3), over the terms defined in the main ontology $\mathcal{P}$. However, it is not expected that the meaning of the terms defined in $\mathcal{Q}$ changes as a consequence of the import since these terms are supposed to be completely specified within $\mathcal{Q}$. Such a side effect would be highly undesirable for the modeling of $\mathcal{P}$ since the ontology engineer of $\mathcal{P}$ might not be an expert on the subject of $\mathcal{Q}$ and is not supposed to alter the meaning of the terms defined in $\mathcal{Q}$ not even implicitly. Nevertheless, it is possible that the meaning of the reused terms could inadvertently change during the import process, potentially due to modeling errors.

In order to illustrate such a situation, suppose that the ontology engineer has learned about the concepts Cystic_Fibrosis and Genetic_Disorder from the ontology $\mathcal{Q}$ (including the axiom α and has decided to introduce additional axioms formalizing the following statements:

"Every instance of Project is different from every instance of Genetic_Disorder and every instance of Cystic_Fibrosis" (5)

"Every project that has_Focus on Cystic_Fibrosis, also has_Focus on Genetic_Disorder" (6)

Note that the statements (5) and (6) can be thought of as adding new information about projects and, intuitively, they should not change or constrain the meaning of the medical terms. Suppose that the ontology engineer has formalized the statements (5) and (6) in ontology using axioms E1 and E2 respectively. At this point, the ontology engineer has introduced modeling errors and, as a consequence, axioms E1 and E2 do not correspond to (5) and (6). E1 actually formalizes the following statement:

"Every instance of Project is different from every common instance of Genetic_Disorder and Cystic_Fibrosis" (7)

and E2 expresses that:

"Every project that either has_Focus on nothing or has_Focus on Cystic_Fibrosis, also has_Focus on Genetic_Disorder" (8)

These kinds of modeling errors are difficult to detect, especially when they do not cause inconsistencies in the ontology. Note that, while axiom E1 does not correspond to (5), it is still a consequence of (5), which means that it should not constrain the meaning of the medical terms. On the other hand, E2 is not a consequence of (6), and, in fact, it constrains the meaning of medical terms. Indeed, the axioms E1 and E2 together with axioms P1—P4 from $\mathcal{P}$ imply new axioms about the concepts Cystic_Fibrosis and Genetic_Disorder, namely their disjointness:

$$\mathcal{P} \models \gamma := \mathsf{Cystic\_Fibrosis} \sqcap \mathsf{Genetic\_Disorder} \sqsubseteq \bot \tag{9}$$

The entailment (9) can be proved using axiom E2 which is equivalent to:

$$\top \sqsubseteq \exists \mathsf{has\_Focus}.(\neg \mathsf{Cystic\_Fibrosis} \sqcup \mathsf{Genetic\_Disorder}) \tag{10}$$

The inclusion (10) and P4 imply that every element in the domain must be a project—that is, $\mathcal{P} \models \top \sqsubseteq \mathsf{Project}$). Now, together with axiom E1, this implies (9). The axioms E1 and E2 not only imply new statements about the medical terms, but also cause inconsistencies when used together with the imported axioms from $\mathcal{Q}$. Indeed, from (1) and (9) we obtain:

$$\mathcal{P} \cup \mathcal{Q} \models \delta := \mathsf{Cystic\_Fibrosis} \sqsubseteq \bot \tag{11}$$

which expresses the unsatisfiability of the concept Cystic _Fibrosis.

In conclusion, it has been observed that importing an external ontology can lead to undesirable side effects in our knowledge reuse scenario. These consequences may appear as the entailment of new axioms or even the occurrence of unsatisfiability issues involving the reused vocabulary.

Thus, an important requirement for the reuse of an ontology $\mathcal{Q}$ within an ontology $\mathcal{P}$ should be that $\mathcal{P} \cup \mathcal{Q}$ produces exactly the same logical consequences over the vocabulary of $\mathcal{Q}$ as $\mathcal{Q}$ alone does. This requirement can be naturally formulated using the well-known notion of a conservative extension, which has recently been introduced in our lectures.

Definition 1 (Deductive Conservative Extension) *Let $\mathcal{T}_1$ and $\mathcal{T}_1 \subseteq \mathcal{T}_2$ be two $\mathcal{L}$-TBoxes and Σ a signature over the description logic $\mathcal{L}$. We say that $\mathcal{T}_2$ is a* deductive Σ-conservative extension *of $\mathcal{T}_1$ w.r.t. $\mathcal{L}$, if for every $\mathcal{L}$-axiom α with $sig(\alpha) \subseteq \Sigma$, we have $\mathcal{T}_1 \models \alpha$ iff $\mathcal{T}_2 \models \alpha$. We say that $\mathcal{T}_2$ is a deductive conservative extension of $\mathcal{T}_1$ if $\mathcal{T}_2$ is a deductive Σ-conservative extension of $\mathcal{T}_1$ w.r.t. $\mathcal{L}$ for $\Sigma = sig(\mathcal{T}_1)$.*

In other words, an ontology $\mathcal{T}_2$ is a deductive Σ-conservative extension of $\mathcal{T}_1$ for a signature Σ and language $\mathcal{L}$ iff every logical consequence α of $\mathcal{T}_1$ constructed using the language $\mathcal{L}$ and names only from Σ, is already a logical consequence of $\mathcal{T}_2$; that is, the additional axioms in $\mathcal{T}_2 \backslash \mathcal{T}_1$ do not result into new logical consequences over the vocabulary Σ. Note that if $\mathcal{T}_2$ is a deductive Σ-conservative extension of $\mathcal{T}_1$ w.r.t. $\mathcal{L}$, then $\mathcal{T}_2$ is a deductive Σ'-conservative extension of $\mathcal{T}_1$ w.r.t. $\mathcal{L}$ for every $\Sigma' \subseteq \Sigma$. The notion of a deductive conservative extension can be directly applied to our ontology reuse scenario.

Definition 2 (Safety for a TBox) *Given $\mathcal{L}$-TBoxes $\mathcal{T}$ and $\mathcal{T}'$, we say that $\mathcal{T}$ is* safe *for $\mathcal{T}'$ (or $\mathcal{T}$ imports $\mathcal{T}'$ in a safe way) w.r.t. $\mathcal{L}$ if $\mathcal{T} \cup \mathcal{T}'$ is a deductive conservative extension of $\mathcal{T}'$ w.r.t. $\mathcal{L}$.*

Hence, the first reasoning task relevant to our ontology reuse scenario can be formulated as follows: given $\mathcal{L}$-TBoxes $\mathcal{T}$ and $\mathcal{T}'$, determine if $\mathcal{T}$ is safe for $\mathcal{T}'$ w.r.t. $\mathcal{L}$.

We have shown that, given $\mathcal{P}$ consisting of axioms P1—P4, E1, E2, and $\mathcal{Q}$ consisting of axioms M1—M5 from Figure 1, there exists an axiom $\delta := \text{Cystic\_Fibrosis} \sqsubseteq \bot$ that uses only names in $\text{sig}(\mathcal{Q})$ such that $\mathcal{Q} \not\models \delta$ but $\mathcal{P} \cup \mathcal{Q} \models \delta$. According to Definition 1, this means that $\mathcal{P} \cup \mathcal{Q}$ is not a deductive conservative extension of $\mathcal{Q}$ w.r.t. any language $\mathcal{L}$ in which δ can be expressed (e.g. $\mathcal{L} = \mathcal{ALC}$). It is possible, however, to show that if the axiom E2 is removed from $\mathcal{P}$ then for the resulting ontology $\mathcal{P}_1 = \mathcal{P} \backslash \{\text{E2}\}$, $\mathcal{P}_1 \cup \mathcal{Q}$ is a deductive conservative extension of $\mathcal{Q}$. The following notion is useful for proving deductive conservative extensions:

Definition 3 (Model Conservative Extension) *Let $\mathcal{T}_1$ and $\mathcal{T}_1 \subseteq \mathcal{T}_2$ be two $\mathcal{L}$-TBoxes and Σ a signature over the description logic $\mathcal{L}$. We say that $\mathcal{T}_2$ is a* model Σ-conservative extension *of $\mathcal{T}_1$, if for every model $\mathcal{I}_1$ of $\mathcal{T}_1$, there exists a model $\mathcal{I}_2$ of $\mathcal{T}_2$ such that $\mathcal{I}_1|_\Sigma = \mathcal{I}_2|_\Sigma$, i.e., the extensions of concept and role names from Σ coincide in $\mathcal{I}_1$ and $\mathcal{I}_2$. We say that $\mathcal{T}_2$ is a model conservative extension of $\mathcal{T}_1$ if $\mathcal{T}_2$ is a model Σ-conservative extension of $\mathcal{T}_1$ for $\Sigma = sig(\mathcal{T}_1)$.*

Observe that the definition of conservative extension provided in the lectures (Definition 6.3) does not impose the requirement that $\mathcal{T}_1$ must be a subset of $\mathcal{T}_2$. Therefore, Definition 6.3 can be considered a generalization of the definition provided in Definition 3 above. The conditions (1) and (2) in Definition 6.3 are immediate consequences of the fact that $\mathcal{T}_1$ is a subset of $\mathcal{T}_2$. Definition 3 specifically states that the axioms in $\mathcal{T}_2 \backslash \mathcal{T}_1$ do not affect the semantics of the names in Σ. It is important to note that $\mathcal{T}_1$ can include additional names beyond those in Σ, and their semantics may be subject to change.

The notion of model conservative extension in Definition 3 can be seen as a semantic counterpart for the notion of deductive conservative extension in Definition 1: the latter is defined in terms of "logical consequences", whereas the former is defined in terms of "models". Intuitively, a TBox $\mathcal{T}_2$ is a model Σ-conservative

extension of $\mathcal{T}_1$ if for every model of $\mathcal{T}_1$ one can find a model of $\mathcal{T}_2$ over the same domain which interprets the names from Σ in the same way. The notion of model conservative extension, however, does not provide a complete characterization of deductive conservative extensions, as given in Definition 1; that is, this notion can be used for proving that an ontology is a deductive conservative extension of another, but not vice versa.

- Show that for every two $\mathcal{L}$-TBoxes $\mathcal{T}_2$, $\mathcal{T}_1 \subseteq \mathcal{T}_2$, and a signature Σ over $\mathcal{L}$, if $\mathcal{T}_2$ is a model Σ-conservative extension of $\mathcal{T}_1$, then it is a also deductive Σ-conservative extension of $\mathcal{T}_1$.
- Show that there exists two $\mathcal{ALC}$-TBoxes $\mathcal{T}_2$, $\mathcal{T}_1 \subseteq \mathcal{T}_2$ such that $\mathcal{T}_2$ is a deductive Σ-conservative extension of $\mathcal{T}_1$ but it is NOT a model conservative extension of $\mathcal{T}_1$.
- Consider the TBox $\mathcal{P}_1$ consisting of the axioms P1—P4 and E1 and the TBox $\mathcal{Q}$ consisting of the axioms M1—M5 from Figure 1. Show that $\mathcal{P}_1 \cup \mathcal{Q}$ is a deductive conservative extension of $\mathcal{Q}$.

Question 9. Σ-Reducts in Acyclic $\mathcal{EL}$
Given an acyclic $\mathcal{EL}$ ontology $\mathcal{T}_2$ and a signature $\Sigma \subseteq \text{sig}(\mathcal{T}_2)$, we say that an $\mathcal{EL}$ TBox $\mathcal{T}_1$ is a Σ-reduct of $\mathcal{T}_2$ if the following conditions hold:

- $\text{sig}(\mathcal{T}_1) \subseteq \Sigma$;
- for any $\mathcal{EL}$-concepts C and D with $\text{sig}(C) \subseteq \Sigma$ and $\text{sig}(D) \subseteq \Sigma$, $\mathcal{T}_1 \models C \sqsubseteq D$ iff $\mathcal{T}_2 \models C \sqsubseteq D$.

Observe that $\mathcal{T}_2$ is a deductive Σ-conservative extension of $\mathcal{T}_1$.

- Does there always exist a Σ-reduct $\mathcal{T}_1$ for any given acyclic $\mathcal{EL}$ ontology $\mathcal{T}_2$ and any $\Sigma \subseteq \text{sig}(\mathcal{T}_2)$?
- If it does, can you develop a terminating, sound, and complete algorithm for computing it? Additionally, provide proof for its termination, soundness, and completeness.

Hint: I encourage you to delve into the intricacies of Section 6.1 of the reference book as it holds the potential to ignite your creativity. Computing $\mathcal{T}_1$ directly from the given $\mathcal{T}_2$ and Σ can be quite challenging. Why not take a step-by-step approach? For example, assume $\Sigma = \text{sig}(\mathcal{T}_2) \backslash \{A\}$, for A a concept name, and then leverage the power of transitivity... In order to accomplish this, it is essential to identify a suitable normal form, develop a corresponding normalization algorithm, and establish its termination, soundness, and completeness (sometimes termination is self-evident). Then, you can proceed to develop an algorithm for computing the Σ-reduct based on this normal form, while ensuring its termination, soundness, and completeness. Recursive calls to this algorithm can be made to handle more complex cases.

Question 10 (with 1 bonus mark). Simpleness of ABox
Consider using arbitrary ABoxes instead of simple ones as input for the task of ontology-mediated querying. Does this affect the data complexity results?

Question 11 (with 1 bonus mark). K-Colorability
Can we establish coNP-hardness of conjunctive query entailment (CQ-entailment) in $\mathcal{ALC}$ w.r.t. data complexity by reducing it to non-k-colorability in graphs? What if k is fixed?

Question 12 (with 1 bonus mark). CW3: Populating Family History
A dataset describing aspects of family history constitutes the content of this question. It contains information about people and the occupations or roles they played. Each individual described has associated information:

- Surname

- Married Surname (if known)
- Birth Year (if known)
- Given Name (first name)

In addition, there may be one or more roles/occupations associated with the individual. Each role/occupation will have

- Year (if known)
- Source
- Occupation (if known — this may be “none given”, stating that the role is unknown)

Data is provided in CSV format, with values separated by commas. The accompanying code is designed to parse this CSV file and create a collection of Bean Objects, complete with appropriate get and set methods. Each row in the spreadsheet corresponds to a Bean object in the collection. It is assumed that the combination of given name, surname, and date of birth is sufficient to uniquely identify individuals. Rows without information on year, source, or occupation can be disregarded.

For this task, you are required to develop a Java program that utilizes the OWL API version 5.0.0 to populate an ontology with the provided data. The ontology, which contains class and property definitions, is included in the archive. No additional classes, properties, or definitions need to be added to the ontology — your modifications should focus solely on individuals. Each person mentioned in the data should be represented as a named individual of type "Person" in the ontology. Each role played should be associated with an instance of "RolePlayed," with appropriate relationships to a role instance, a source, and a year (if known). The individuals representing roles played can be given specific names or left as anonymous individuals, depending on your preference. Note that the role classes in the CSV file should align with the role hierarchy in the ontology, although some processing may be required to map roles/occupations in the data to the corresponding roles in the ontology. Additionally, special consideration should be given to the case of "none given," which is not a role name but rather indicates the absence of an explicit role.

Your submission will undergo evaluation by loading a distinct dataset and conducting queries on it. A sample dataset is provided in the "test/test.csv" file, and the ontology for population can be accessed in "resources/ontology.owl." Within the owlapi.khkb.fspopulation.CW3 class, you are required to implement three specific methods:

- loadOntology(OWLOntologyManager manager, IRI ontologyIRI)
  Loading ontology from physical IRI
- populateOntology (OWLOntologyManager manager, OWLOntology ontology, Collection<JonDataBean> beans)
  Populating the ontology using the collection of Java beans holding the data from CSV
- saveOntology (OWLOntologyManager manager, OWLOntology ontology, IRI locationIRI)
  Saving the ontology back to the disk

Feel free to include additional methods in this class, but please be sure that you do not alter the signature of the existing methods or the no-argument constructor. All the necessary functionality must be encapsulated within CW3.java. As this is the only file you will submit, please do not implement any essential functionality outside of CW3.java. The harness is configured as a Maven project, and you can conduct your testing using the three implemented JUnit tests. To set up your project, follow these steps:

1. Extract the cw3.zip somewhere on your computer
2. Open your Eclipse, select File->Import->Existing Maven Project and select the directory you just unpacked.
3. Go to the Build Path settings and make sure that the JRE selected is in fact the one from your JDK.
4. Right-click on the project: Maven clean.
5. Right-click on the project: Maven install

You will see your project failing to build because it does not pass the Junit tests. Now you are good to go. As a recommendation, I suggest running the JUnit tests regularly within Eclipse to track your progress. These tests will utilize the input from resources/test.csv, the ontology from resources/ontology.owl, and generate a populated ontology in fhkb-ai.owl. When you have completed the assignment, kindly submit a zip archive containing the CW3.java file located in your project directory.

Question 13 (with 1 bonus mark). CW5: Implementing Ontology Queries

CW5 is all about querying implementing and an OWL-based querying system. Your task will be to query the reasoner for sub-classes, equivalent classes, and instances, and store the results in a QueryResult object. In the Java class CW5 (which follows the same project layout as CW3), you are required to implement the following methods. Please refrain from modifying any class other than CW5, as this is the only class you are allowed to submit. To test your implementation, you can exploit the provided classes App.java and PizzaOrderingSystemApp.java, which offer a potentially enjoyable way to verify the correctness of your code. Similar to running the unit tests in CW3, you can launch these classes in Eclipse by clicking the small white and green arrow icon when the class is open. Alternatively, if you prefer a more straightforward approach, you can rely on the pre-implemented unit tests to assess your progress. It is important to note that passing the unit tests does not guarantee a perfect mark this time. We will employ additional tests to further evaluate your solution. Once you have completed the assignment, please compress CW5.java into a zip file and submit it via the PedagogySquare platform. Good luck!

1. public Set<QueryResult> performQuery (OWLClassExpression exp, Query Type type) {...}
   This method takes in an arbitrary expression in OWL, like “Person and hasBirthYear value 1964” or “hasTopping some MeatTopping” and queries the ontology for the following three types of knowledge:
   EquivalentClasses: you are asked to return all those (named) classes that are equivalent to the expression (exp), using the (already fully initialized) reasoner.
   Sub-classes: return all those classes that are (named) sub-classes of the expression, using the reasoner.
   Instances: return all those individuals that are instances of the expression, using the reasoner.
   Query results are stored in query result objects. These are created for example as follows:
   QueryResult qr = new QueryResult(ind, false, type);
   Note that you need to include the information whether the inference is direct or indirect. Example: Given three classes with A subclass B subclass C, then A is a direct subclass of B and B is a direct subclass of C, but A is an indirect subclass of C (through B!). In order to solve this problem, you will query the reasoner for direct and indirect sub-class separately. After creating the QueryResult, add it to the provided set.
2. public Boolean isValidPizza (OWLClassExpression exp) {...};
   In this method, you check whether the supplied class expression exp is a valid pizza expression, i.e., whether it is inferred to be a Pizza.

3. public Set<QueryResult> filterNamedPizzas (Set<QueryResult> results) {...};
   This question is similar to the one before, only that you are asked to filter from a set of results those that correspond to NamedPizza's, such as Margherita or AmericanHot.

4. public Set<OWLClassExpression> getAllSuperClassExpressions (OWLClass ce) {...};

This question requires a bit of thinking. You are asked to query the ontology to gather all available information about the class ce. In order to get an idea of what "AL" means, you can open Protégé and look at the "Classe" tab. Clicking on a class will reveal its super-classes, equivalent-classes, as well as inherited super-classes (Anonymous Ancestors) and indirect super-classes. Unfortunately, the reasoner will not easily give you access to those. In order to get full marks for this task, it is sufficient to query for all super-classes (direct and indirect, using the reasoner), and somehow find a way to obtain the anonymous super-classes using the ontology directly (that will need a bit of thought, do not despair too quickly). A perfect solution will test, for all sub-expressions in the ontology, whether ce is a sub-concept of it. Attempt this only if you are really confident with the OWL API by now.