Logic-based Ontology Engineering

Topics

We discuss:

- Debugging ontologies: axiom pinpointing;
- Modular ontologies;
- Versioning for ontologies;
- Module extraction;

Debugging Ontologies using Axiom Pinpointing

One important way of checking the quality of an ontology $\mathcal{O}$ is to check, for a number of inclusions $C \sqsubseteq D$ that **should not** follow from $\mathcal{O}$, that they, indeed, do not follow from $\mathcal{O}$.

For example, in this way it has been discovered that a certain version of SNOMED CT implied

$$F = (\textbf{Amputation\_of\_Finger} \sqsubseteq \textbf{Amputation\_of\_Hand})$$

Of course, F should not follow from SNOMED CT and so, in order to revise SNOMED CT, one would like to automatically generate an **explanation** as to why F followed from SNOMED CT.

Minimal subsets (pinpointing sets) of SNOMED CT from which F follows provide such an explanation.

Definition of Axiom Pinpointing

Let $\mathcal{T}$ be an ontology and C, D concepts such that

$$\mathcal{T} \models C \sqsubseteq D.$$

The **pinpointing set**

$$\mathbf{Pin}(\mathcal{T}, C \sqsubseteq D)$$

of $\mathcal{T}$ w.r.t. $C \sqsubseteq D$ consists of all minimal subsets $\mathcal{T}'$ of $\mathcal{T}$ such that

$$\mathcal{T}' \models C \sqsubseteq D.$$

Example

$$\mathcal{T} = \{\mathbf{Father} \sqsubseteq \mathbf{Male}, \mathbf{Male} \sqsubseteq \mathbf{Human}, \mathbf{Car} \sqsubseteq \mathbf{Vehicle}\}$$

Then $\mathbf{Pin}(\mathcal{T}, \mathbf{Father} \sqsubseteq \mathbf{Human}\}$ consists of only one set, namely the set

$$\{\mathbf{Father} \sqsubseteq \mathbf{Male}, \mathbf{Male} \sqsubseteq \mathbf{Human}\}$$

More examples

Let

$$\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq E, A \sqsubseteq F, F \sqsubseteq E, C \sqsubseteq D\}$$

Then $\mathcal{T} \models A \sqsubseteq E$.

There are two sets in $\mathbf{Pin}(\mathcal{T}, A \sqsubseteq E\}$, namely

$$\{A \sqsubseteq B, B \sqsubseteq E\}$$

and

$$\{A \sqsubseteq F, F \sqsubseteq E\}.$$

More Examples: NCI

Let NCI be the following TBox:

$$\begin{aligned}
\mathbf{Cystic_Fibrosis} &\equiv \mathbf{Fibrosis} \sqcap \exists \mathbf{located_In.Pancreas} \sqcap \exists \mathbf{has_Origin.Genetic_Origin} \\
\mathbf{Genetic_Fibrosis} &\equiv \mathbf{Fibrosis} \sqcap \exists \mathbf{has_Origin.Genetic_Origin} \\
\mathbf{Genetic_Fibrosis} &\sqsupseteq \mathbf{Fibrosis} \sqcap \exists \mathbf{located_In.Pancreas} \\
\mathbf{Genetic_Fibrosis} &\sqsubseteq \mathbf{Genetic_Disorder} \\
\mathbf{DEFBI_Gene} &\sqsubseteq \mathbf{Immuno_Protein_Gene} \sqcap \exists \mathbf{associated_with.Cystic_Fibrosis}
\end{aligned}$$

NCI

There are two sets in **Pin**(NCI, **Cystic_Fibrosis** $\sqsubseteq$ **Genetic_Disorder**):

$$\begin{aligned}
\textbf{Cystic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\sqsubseteq \textbf{Genetic_Disorder}
\end{aligned}$$

and

$$\begin{aligned}
\textbf{Cystic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\sqsupseteq \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \\
\textbf{Genetic_Fibrosis} &\sqsubseteq \textbf{Genetic_Disorder}
\end{aligned}$$

More examples

Consider the TBox consisting of the axioms

a $\mathbf{Human} \sqsubseteq \exists \mathbf{child\_of}.\mathbf{Human}$;

b $\mathbf{Human} \sqsubseteq \mathbf{Living\_Being}$;

c $\exists \mathbf{child\_of}.\mathbf{Living\_Being} \sqsubseteq \mathbf{Has\_Birthday}$;

d $\mathbf{Living\_Being} \sqsubseteq \mathbf{Has\_Birthday}$.

Then $\mathcal{T} \models \mathbf{Human} \sqsubseteq \mathbf{Has\_Birthday}$. The set $\mathbf{Pin}(\mathcal{T}, \mathbf{Human} \sqsubseteq \mathbf{Has\_Birthday})$ consists of

$$\{b, d\}$$

and

$$\{a, b, c\}$$

Algorithmic problems

- There can be exponentially many sets in $\mathbf{Pin}(\mathcal{T}, C \sqsubseteq D)$;
- Testing whether there exists a member of $\mathbf{Pin}(\mathcal{T}, C \sqsubseteq D)$ of size $\leq n$ is non-tractable, even for $\mathcal{EL}$;
- However, computing ONE member of $\mathbf{Pin}(\mathcal{T}, C \sqsubseteq D)$ is tractable, given an oracle for subsumption (see next slide).
- Hence, computing ONE member of $\mathbf{Pin}(\mathcal{T}, C \sqsubseteq D)$ for $\mathcal{T}$, C and D in $\mathcal{EL}$ is tractable.

An algorithm for finding ONE member of Pin$(\mathcal{T}, C \sqsubseteq D)$

Input $\mathcal{T} = \{\alpha_1, \ldots, \alpha_n\}$ and $C \sqsubseteq D$.

1. if $C \not\sqsubseteq_{\mathcal{T}} D$, then
2. return **Pin**$(\mathcal{T}, C \sqsubseteq D)$ empty
3. set $\mathcal{S} := \mathcal{T}$
4. for $1 \leq i \leq n$ do
5. if $C \sqsubseteq_{\mathcal{S} \setminus \{\alpha_i\}} D$ then
6. $\mathcal{S} := \mathcal{S} \setminus \{\alpha_i\}$
7. return $\mathcal{S}$.

Modularity

Ontologies should not be monolithic entities, but should have **modular structure**.

Currently, a lot of ongoing research on how such a modular structure of an ontology should be defined. For example, there is an "import another ontology" construct in the OWL standard, but it has no semantics.

One important property of modules in general is that they should

"function independently"

from the system they are part of. We briefly explore what this means in the context of ontologies. For simplicity, we mostly consider $\mathcal{EL}$ TBoxes only.

Signatures

- A **signature** is a finite set of concept and role names.
- The **signature** $\mathbf{sig}(C)$ of a concept C is the set of concept and role names that occur in C. For example,

$$\mathbf{sig}(\mathbf{Human} \sqcap \exists \mathbf{has\_child}.\top) = \{\mathbf{Human}, \mathbf{has\_child}\}$$

- The **signature** $\mathbf{sig}(C \sqsubseteq D)$ of a concept inclusion is defined as

$$\mathbf{sig}(C \sqsubseteq D) = \mathbf{sig}(C) \cup \mathbf{sig}(D).$$

- The **signature** $\mathbf{sig}(\mathcal{T})$ of a TBox $\mathcal{T}$ is defined as the union of the signatures of its concept inclusions.

Intuitively, the signature $\mathbf{sig}(\mathcal{T})$ summarises the **subject matter** or topic of $\mathcal{T}$. In contrast, the symbols $\sqcap$, $\exists$, and $\top$ are logical symbols that are not part of the subject matter of $\mathcal{T}$.

Modules

Let $\mathcal{T}$ be a TBox. A subset $\mathcal{M}$ of $\mathcal{T}$ is called a **module** of $\mathcal{T}$ if

$$\mathcal{M} \models C \sqsubseteq D \quad \Leftrightarrow \quad \mathcal{T} \models C \sqsubseteq D$$

for all concept inclusions $C \sqsubseteq D$ with $\mathbf{sig}(C \sqsubseteq D) \subseteq \mathbf{sig}(\mathcal{M})$.

Thus,

- a module "functions independently" from the TBox in the sense that it implies the same concept inclusions for its own subject matter as the whole TBox.
- Moreover, the TBox does not "interfere" with the module in that it does not influence the meaning that the module defines for its own terms.

A EU ontology using a medical ontology as a module

Consider an ontology $\mathcal{EU}$ about EU research projects that uses as a module for its medical terms a medical ontology:

$$\begin{aligned}
\textbf{Genetic_Disorder_Project} &\equiv \textbf{Project} \sqcap \exists \textbf{has_Focus.Genetic_Disorder} \\
\textbf{Cystic_Fibrosis_EUProject} &\sqsubseteq \textbf{EUProject} \sqcap \exists \textbf{has_Focus.Cystic_Fibrosis} \\
\textbf{EUProject} &\sqsubseteq \textbf{Project}
\end{aligned}$$

uses NCI (i.e., NCI $\subseteq \mathcal{EU}$):

$$\begin{aligned}
\textbf{Cystic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\sqsupseteq \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \\
\textbf{Genetic_Fibrosis} &\sqsubseteq \textbf{Genetic_Disorder} \\
\textbf{DEFBI_Gene} &\sqsubseteq \textbf{Immuno_Protein_Gene} \sqcap \exists \textbf{associated_with.Cystic_Fibrosis}
\end{aligned}$$

Discussion

- The signature $\mathbf{sig}$(NCI) is given by: Cystic_Fibrosis, Fibrosis, located_In, Pancreas, has_Origin, Genetic_Origin, Genetic_Disorder, DEFBI_Gene, Immuno_Protein_Gene, associated_with.
- The signature of $\mathcal{EU}$ contains $\mathbf{sig}$(NCI) and Genetic_Disorder_Project, Project, has_Focus, Cystic_Fibrosis_EUProject, EUProject.

NCI is a **module** of $\mathcal{EU}$ because

$$\text{NCI} \models \boldsymbol{C} \sqsubseteq \boldsymbol{D} \quad \Leftrightarrow \quad \mathcal{EU} \models \boldsymbol{C} \sqsubseteq \boldsymbol{D}$$

for all concept inclusions using medical terms only (i.e., all $\boldsymbol{C} \sqsubseteq \boldsymbol{D}$ such that $\mathbf{sig}(\boldsymbol{C} \sqsubseteq \boldsymbol{D}) \subseteq \mathbf{sig}(\text{NCI})$).

Proof that NCI is a module of $\mathcal{EU}$

As $\mathcal{EU} \supseteq$ NCI, we clearly have

$$\text{NCI} \models C \sqsubseteq D \quad \Rightarrow \quad \mathcal{EU} \models C \sqsubseteq D,$$

for **all** $C \sqsubseteq D$.

Conversely, we show

$$\text{NCI} \not\models C \sqsubseteq D \quad \Rightarrow \quad \mathcal{EU} \not\models C \sqsubseteq D,$$

for all $C \sqsubseteq D$ with $\mathbf{sig}(C \sqsubseteq D) \subseteq \mathbf{sig}(\text{NCI})$.

Assume NCI $\not\models C \sqsubseteq D$ for a $C \sqsubseteq D$ with $\mathbf{sig}(C \sqsubseteq D) \subseteq \mathbf{sig}(\text{NCI})$.
We show that $\mathcal{EU} \not\models C \sqsubseteq D$.

We find an interpretation $\mathcal{I}$ satisfying NCI with a $d \in \Delta^{\mathcal{I}}$ such that

$$d \in C^{\mathcal{I}}, \quad d \notin D^{\mathcal{I}}$$

Define an extension $\mathcal{I}'$ of $\mathcal{I}$ by setting

- $\textbf{Project}^{\mathcal{I}'} = \textbf{EUProject}^{\mathcal{I}'} \subseteq \Delta^{\mathcal{I}}$ arbitrary;
- $\textbf{has\_Focus}^{\mathcal{I}'} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ arbitrary;
- $\textbf{Genetic\_Disorder\_Project}^{\mathcal{I}'} = (\textbf{Project} \sqcap \exists \textbf{has\_Focus.Genetic\_Disorder})^{\mathcal{I}'}$
- $\textbf{Cystic\_Fibrosis\_EUProject}^{\mathcal{I}'} = (\textbf{EUProject} \sqcap \exists \textbf{has\_Focus.Cystic\_Fibrosis})^{\mathcal{I}'}$.

By definition, $\mathcal{I}'$ satisfies $\mathcal{EU}$.

Moreover, we still have

$$d \in C^{\mathcal{I}'}, \quad d \notin D^{\mathcal{I}'}$$

because $\textbf{sig}(C \sqsubseteq D) \subseteq \textbf{sig}(\text{NCI})$. It follows that $\mathcal{EU} \not\models C \sqsubseteq D$.

General Result

Let $\mathcal{M} \subseteq \mathcal{O}$ be such that $\mathcal{O} \setminus \mathcal{M}$ is an

- acyclic $\mathcal{EL}$ terminology such that no A with $A \equiv C$ or $A \sqsubseteq C$ in $\mathcal{O} \setminus \mathcal{M}$ is in $\mathbf{sig}(\mathcal{M})$.

Then $\mathcal{M}$ is a module of $\mathcal{O}$.

Proof as above: Every interpretation $\mathcal{I}$ satisfying $\mathcal{M}$ can be expanded to an interpretation $\mathcal{I}'$ satisfying $\mathcal{O}$ by interpreting the new symbols of $\mathcal{O}$ according to the definitions in $\mathcal{O} \setminus \mathcal{M}$.

In mathematical logic such an $\mathcal{O}$ is called a **definitorial extension** of $\mathcal{M}$.

$\mathcal{EU} \setminus$ NCI is not a module of $\mathcal{EU}$

NCI influences (and should influence) the meaning of terms in $\mathcal{EU} \setminus$ NCI. Firstly, we have

$$\text{NCI} \models \textbf{Cystic\_Fibrosis} \sqsubseteq \textbf{GeneticDisorder}$$

and so

$$\mathcal{EU} \models \textbf{Cystic\_Fibrosis} \sqsubseteq \textbf{GeneticDisorder}$$

But on the other hand,

$$\mathcal{EU} \setminus \text{NCI} \not\models \textbf{Cystic\_Fibrosis} \sqsubseteq \textbf{GeneticDisorder}$$

and we obtain

$$\mathcal{EU} \setminus \text{NCI} \not\models \textbf{Cystic\_Fibrosis\_EUProject} \sqsubseteq \textbf{Genetic\_Disorder\_Project}$$

and

$$\mathcal{EU} \models \textbf{Cystic\_Fibrosis\_EUProject} \sqsubseteq \textbf{Genetic\_Disorder\_Project}$$

Discussion of Complexity

Module checking is the problem of deciding whether a subset $\mathcal{M}$ of an ontology $\mathcal{O}$ is a module.

Module checking is typically much harder than satisfiability or subsumption checking. For example,

- For $\mathcal{EL}$ this problem is ExpTime-complete;
- For $\mathcal{ALC}$ it is 2ExpTime-complete;
- For $\mathcal{ALCQIO}$ it is undecidable.

$\mathcal{EL}$ terminologies are an exception. For them "module-checking" is tractable.

A variety of heuristic approaches to the definition of modules have been developed as well.

Versioning

Ontologies constantly evolve; they are updated, corrected, and refined. Older versions are kept for comparison with updates, referencing, tracing the originis of definitions, and audit purposes.

Dealing with multiple versions of the same information unit is a standard problem in CS and version control is a well established technology. Modern version control systems provide a range of operations including support for collaborative development, branching, merging, etc.

We briefly discuss a novel challenge for ontology versioning:

"what is the difference between two ontologies?"

Currently Existing Support

The W3C standard OWL introduces a number of statements related to versioning (for example, `owl:versionInfo`, `owl:backwardCompatibleWith`, etc). However, they do not have a formal semantics and a **principled** approach to comparing distinct versions of an ontology is regarded as an important research problem.

One can distinguish three approaches to describing the difference between ontologies:

1. versioning based on syntactic difference (syntactic diff);
2. versioning based on structural difference (structural diff);
3. versioning based on logical difference (logical diff).

Syntactic Diff

- The syntactic diff underlies most existing version control systems used in software development. It works with text files and represents the difference between versions as blocks of text present in one version but not another, ignoring any meta-information about the document.

- Ontology versioning cannot rely on a purely syntactic diff operation: a variety of syntactic differences, for example, the order of ontology axioms or existence of redundant axioms, does not affect the semantics of ontologies.

  For example,

  $$\mathcal{T} = \{A \sqsubseteq B, E \sqsubseteq F\}$$

  is the same ontology as

  $$\mathcal{T}' = \{E \sqsubseteq F, A \sqsubseteq B\}$$

  but they are syntactically different.

Structural Diff

- The structural diff extends the syntactic diff by taking into account structural meta-information about the distinct versions of files compared.
- It has been suggested for dealing with structured and hierarchical documents such as UML diagrams, database schemas, or XML documents.
- For ontologies, the main characteristic of the structural diff is that it regards them as structured objects, such as a concept hierarchy or a set of concept definitions.

  For example,

$$\mathcal{T} = \{\mathbf{Father} \sqsubseteq \mathbf{Human} \sqcap \mathbf{Male}, \mathbf{Father} \sqsubseteq \exists \mathbf{has\_child}.\top\}$$

  is regarded as different from

$$\mathcal{T}' = \{\mathbf{Father} \sqsubseteq \mathbf{Human} \sqcap \mathbf{Male} \sqcap \exists \mathbf{has\_child}.\top\}$$

  although they are logically equivalent.

Structural Diff

- Changes to ontologies are described in terms of structural operations, for example, adding or deleting a definition, extending a concept definition, moving a concept from one place in the hierarchy to another, adding or deleting an inclusion, etc.

  Example: $\mathcal{T}'$ is obtained from $\mathcal{T}$ by expanding the first inclusion

  $$\textbf{Father} \sqsubseteq \textbf{Human} \sqcap \textbf{Male}$$

  to

  $$\textbf{Father} \sqsubseteq \textbf{Human} \sqcap \textbf{Male} \sqcap \exists \textbf{has\_child.} \top$$

  and then removing

  $$\textbf{Father} \sqsubseteq \exists \textbf{has\_child.} \top$$

- Most current ontology editors and ontology management systems such as Protege, SWOOP, OBO-Edit, and OntoView support ontology versioning based on structural diff.

Syntax Independence

Two ontologies, $\mathcal{O}$ and $\mathcal{O}'$ are **logically equivalent** if, and only if, for all interpretations $\mathcal{I}$: $\mathcal{I}$ is a model of $\mathcal{O}$ if, and only if, $\mathcal{I}$ is a model of $\mathcal{O}'$.

Example: the ontologies $\mathcal{O}$ and $\mathcal{O}'$ with

$$\mathcal{O} = \{\top \sqsubseteq \forall r.A\}, \quad \mathcal{O}' = \{\exists r^{-}.\top \sqsubseteq A\}$$

are logically equivalent.

An operation diff is **syntax independent** iff

$$\text{diff}(\mathcal{O}_1, \mathcal{O}_2) = \text{diff}(\mathcal{O}'_1, \mathcal{O}'_2)$$

whenever $\mathcal{O}_1$ and $\mathcal{O}'_1$ as well as $\mathcal{O}_2$ and $\mathcal{O}'_2$ are logically equivalent.

The structural diff is syntax dependent!

Logical Diff

Difference should be always given relative to a subject matter. Let $\boldsymbol{S}$ be a signature (subject matter). The logical difference between two ontologies versions $\mathcal{O}_1$ and $\mathcal{O}_2$ of an ontology with respect to $\boldsymbol{S}$,

$$\textbf{Diff}_S(\mathcal{O}_1, \mathcal{O}_2),$$

consists of the set of subsumptions $C \sqsubseteq D$ with $\textbf{sig}(C \sqsubseteq D) \subseteq S$ which follow from $\mathcal{O}_1$ but not from $\mathcal{O}_2$, or vice versa:

$$\begin{aligned}\textbf{Diff}_S(\mathcal{O}_1, \mathcal{O}_2) \;=\; & \{C \sqsubseteq D \mid \mathcal{O}_1 \models C \sqsubseteq D, \mathcal{O}_2 \not\models C \sqsubseteq D, \textbf{sig}(C \sqsubseteq D) \subseteq S\} \cup \\ & \{C \sqsubseteq D \mid \mathcal{O}_2 \models C \sqsubseteq D, \mathcal{O}_1 \not\models C \sqsubseteq D, \textbf{sig}(C \sqsubseteq D) \subseteq S\}\end{aligned}$$

From a logical viewpoint, $\mathcal{O}_1$ and $\mathcal{O}_2$ say the same about S if, and only if, $\textbf{Diff}_S(\mathcal{O}_1, \mathcal{O}_2) = \emptyset$.

Note: a set $\mathcal{M} \subseteq \mathcal{O}$ is a module if, and only if $\textbf{Diff}_S(\mathcal{M}, \mathcal{O}) = \emptyset$.

Problems to be solved for Logical Diff

- If $\mathbf{Diff}_S(\mathcal{O}_1, \mathcal{O}_2) \neq \emptyset$, then it is mostly infinite. For example, for

$$\mathcal{T}_1 = \mathcal{EU}, \quad \mathcal{T}_2 = \mathcal{EU} \setminus \text{NCI}, \quad S = \mathbf{sig}(\mathcal{EU}) \setminus \mathbf{sig}(\text{NCI}),$$

  $\mathbf{Diff}_S(\mathcal{T}_1, \mathcal{T}_2)$ contains

$$\mathbf{Cystic\_Fibrosis\_EUProject} \sqsubseteq \mathbf{Genetic\_Disorder\_Project}$$

  but also infinitely many other inclusions such as

$$\mathbf{Cystic\_Fibrosis\_EUProject} \sqcap \mathbf{Cystic\_Fibrosis\_EUProject} \sqsubseteq \mathbf{Genetic\_Disorder\_Project}$$

  Thus, a finite representation of $\mathbf{Diff}_S(\mathcal{T}_1, \mathcal{T}_2)$ has to be chosen.

- Even deciding whether $\mathbf{Diff}_S(\mathcal{T}_1, \mathcal{T}_2) \neq \emptyset$ is non-tractable (and sometimes undecidable) for standard ontology languages (except $\mathcal{EL}$ terminologies).

A Generalisation of Modules: S-Modules

Let $\mathcal{M} \subseteq \mathcal{O}$ and S a signature. $\mathcal{M}$ is called a

S-module of $\mathcal{O}$

if $\mathbf{Diff}_S(\mathcal{M}, \mathcal{O}) = \emptyset$.

If one is interested in a set of terms S but does not want to use the whole ontology $\mathcal{O}$, one can instead use any S-module of $\mathcal{O}$.

Problem: Extract a **minimal** S-module from a given ontology $\mathcal{O}$ for re-use.

Example

Let S consist of **Cystic_Fibrosis** and **Genetic_Disorder**. Extract a minimal S-module from NCI:

$$\begin{aligned}
\textbf{Cystic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\sqsupseteq \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \\
\textbf{Genetic_Fibrosis} &\sqsubseteq \textbf{Genetic_Disorder} \\
\textbf{DEFBI_Gene} &\sqsubseteq \textbf{Immuno_Protein_Gene} \sqcap \exists \textbf{associated_with.Cystic_Fibrosis}
\end{aligned}$$

The minimal S-modules

There are two minimal S-modules:

$$\begin{aligned}
\textbf{Cystic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\sqsubseteq \textbf{Genetic_Disorder}
\end{aligned}$$

and

$$\begin{aligned}
\textbf{Cystic_Fibrosis} &\equiv \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \sqcap \exists \textbf{has_Origin.Genetic_Origin} \\
\textbf{Genetic_Fibrosis} &\sqsupseteq \textbf{Fibrosis} \sqcap \exists \textbf{located_In.Pancreas} \\
\textbf{Genetic_Fibrosis} &\sqsubseteq \textbf{Genetic_Disorder}
\end{aligned}$$

Algorithm Extracting one minimal S-Module

Input $\mathcal{O} = \{\alpha_1, \ldots, \alpha_n\}$ and S.

1. $\mathcal{M} := \mathcal{O}$
2. for $1 \leq i \leq n$ do
3. if $\mathbf{Diff}_S(\mathcal{M} \setminus \{\alpha_i\}, \mathcal{M}) = \emptyset$,
4. then $\mathcal{M} := \mathcal{M} \setminus \{\alpha_i\}$.
5. return $\mathcal{M}$