A Study on the Correspondence between FCA and $\mathcal{ELI}$ Ontologies

Melisachew Wudage Chekol, Mehwish Alam, and Amedeo Napoli

LORIA (INRIA, CNRS, and Université de Lorraine)
France
{melisachew.chekol,mehwish.alam,amedeo.napoli}@inria.fr

Abstract. The description logic $\mathcal{EL}$ has been used to support ontology design in various domains, and especially in biology and medicine. $\mathcal{EL}$ is known for its efficient reasoning and query answering capabilities. By contrast, ontology design and query answering can be supported and guided within an FCA framework. Accordingly, in this paper, we propose a formal transformation of $\mathcal{ELI}$ (an extension of $\mathcal{EL}$ with *inverse roles*) ontologies into an FCA framework, i.e. $K_{\mathcal{ELI}}$, and we provide a formal characterization of this transformation. Then we show that SPARQL query answering over $\mathcal{ELI}$ ontologies can be reduced to lattice query answering over $K_{\mathcal{ELI}}$ concept lattices. This simplifies the query answering task and shows that some basic semantic web tasks can be improved when considered from an FCA perspective.

1 Introduction

Relying on Semantic Web (SW) languages and principles, several ontologies have been created in various domains, especially, in biology and medicine. In addition to that, since the conception of linked data publishing principles, over 295 linked (open) datasets have been produced[1]. Querying these data is mainly done through the W3C recommended query language SPARQL[2].

In parallel, knowledge discovery in data represented by means of objects and their properties can be done using formal concept analysis (FCA) [9]. Concept lattices can reveal hidden relations within data and can be used for organizing and classifying data. A survey of the benefits of FCA to SW and vice versa has been proposed in [14]. As mentioned in that paper, a few of these benefits ranges from knowledge discovery, ontology completion, to computing subsumption hierarchy of least common subsumers. Additionally, studies in [7] and [12] are based on FCA for managing SW data while finite models of description logics (as $\mathcal{EL}$) are explored in [3,4]. All these studies propose methods to use FCA in the analysis of SW data. Nevertheless, none of them offer a precise way of representing SW data within a formal context. We deem it necessary to provide mathematically founded methods to formalize the representation and the analysis of SW data.

[1] http://linkeddata.org/
[2] http://www.w3.org/TR/sparql11-query/

In this work, we focus particularly on $\mathcal{ELI}$ (an extension of $\mathcal{EL}$ with inverse roles) ontologies. $\mathcal{EL}$ is one of OWL 2 profiles (OWL 2 $\mathcal{EL}$). In fact, OWL 2 $\mathcal{EL}$ is used mainly for designing large biomedical ontologies such as SNOMED-CT[3], and the NCI thesaurus[4]. A common feature of these ontologies is that they possess large concept hierarchies that can be queried with SPARQL. Answering SPARQL queries is done by binding variables of the query into terms of the queried ontology. However, including inferred data in the query answers requires either a reasoner to infer all implicit data or query rewriting using regular expression patterns (that enable navigation in a hierarchy) [10]. The latter obliges the user to know the nuts and bolts of SPARQL. To overcome these difficulties, we reduce SPARQL query answering in $\mathcal{ELI}$ ontologies into query answering in concept lattices along with the transformation of the queried ontology into a formal context. Querying a concept lattice appears to be a less complex task than using SPARQ. Further, the lattice organization, i.e., partial ordering, can help understanding the relations between data and visualization of SW data.

Overall, in this paper, we work towards (i) a formal characterization of the translation of ontologies into a formal context, (ii) minimizing the difficulty of SPARQL query answering over ontologies into LQL (Lattice Query Language) query answering over concept lattices, and finally (iii) providing organization of SPARQL query answers with the use of concept lattices.

Outline: after presenting the basics of $\mathcal{ELI}$, SPARQL and FCA (§2), we show how to transform $\mathcal{ELI}$ ontologies into formal contexts (§3). We then present a query language for concept lattices called LQL (§4). Therefore, SPARQL query answering over $\mathcal{ELI}$ ontologies can be reduced to LQL query answering over $K_{\mathcal{ELI}}$ concept lattices (§5). Finally, we present the related works (§6) along with a summary of concluding remarks (§7).

2 Preliminaries

In this section, we provide a very brief and intuitive introduction of the description logic $\mathcal{ELI}$ and FCA. For a detailed discussion, we refer the readers to [11,2,9,6].

In $\mathcal{ELI}$, classes are inductively defined from a set $\mathrm{N_C}$ of *class* names, a set $\mathrm{N_R}$ of *role* names, and a set $\mathrm{N_I}$ of *individual* names ($\mathrm{N_C}$, $\mathrm{N_R}$, and $\mathrm{N_I}$ are finite), using the constructors: $\top$, $C \sqcap D$, and $\exists R.C$ are classes. Where C and D refer to classes, R refers to a role name or its inverse R^-, and in the assertion $C(a)$, a refers to an individual. In this paper, we consider $\exists R.C$ classes with $C \in \mathrm{N_C}$, i.e, C is an atomic concept in the expression of $\exists R.C$. The TBox of a $\mathcal{EL}$ knowledge base contains a set of class inclusion axioms such as $C \sqsubseteq D$. The ABox contains class and role assertions: $C(a)$ and $R(a, b)$. The semantics of $\mathcal{ELI}$ is broadly discussed in [2]. The semantics of $\mathcal{ELI}$-classes is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ is a non-empty set of

[3] http://www.ihtsdo.org/snomed-ct/
[4] http://ncit.nci.nih.gov/

individuals and the *the interpretation function* $\cdot^{\mathcal{I}}$ maps each class name $A \in \mathrm{N_C}$ to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $R \in \mathrm{N_R}$ to a binary relation $R^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, and each individual name $a \in \mathcal{N_I}$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary class descriptions is defined inductively [2].

SPARQL is a W3C recommended query language [13] based on simple graph patterns. It allows variables to be bound to components in the queried graph. In addition, operators akin to relational joins, unions, left outer joins, selections, and projections can be combined to build more expressive queries. Queries are formed from query patterns which in turn are defined inductively from *path patterns*, i.e., tuple $t \in$ UBV $\times\ e \times$ UBLV, with V a set of variables disjoint from UBL (URIs, Blank nodes and Literals – are used to identify values such as strings, integers and dates.), and e is regular path expression. Path patterns grouped together using operators AND (.) and UNION form *query patterns.*

Definition 1. *A query pattern q is inductively defined as:*

$$q ::= \ \mathrm{UBV} \times e \times \mathrm{UBLV} \mid q_1 \ . \ q_2 \mid \{q_1\} \ \textit{UNION} \ \{q_2\}$$
$$e ::= \ \epsilon \mid U \mid V \mid e_1/e_2 \mid e_1 \mid e_2 \mid e^+ \mid e^*$$

A SPARQL *SELECT* query can be formed according to the following syntax: SELECT W FROM $\mathcal{O}$ WHERE $\{q\}$. The FROM clause identifies the queried ontology $\mathcal{O}$ on which the query will be evaluated, WHERE contains a query pattern q that the query answers should satisfy and SELECT singles out the answer variables $W \in V$ from the query pattern. For this work, we consider only AND and UNION SPARQL queries.

A formal context represents data using objects, attributes, and their relationships. Formally, it is a triple $K = (G, M, I)$ where G a set of objects, M a set of attributes, and $I \subseteq G \times M$ is a relation. A derivation operator ($'$) is used to compute *formal concepts* of a context. Given a set of objects, the operator derives common attributes of these objects and vice versa. A set of formal concepts ordered with the set inclusion relation form a *concept lattice* [6].

In the next section, we show the transformation of $\mathcal{ELI}$ ontologies into formal contexts.

3 Transforming $\mathcal{ELI}$ Ontologies into Formal Contexts

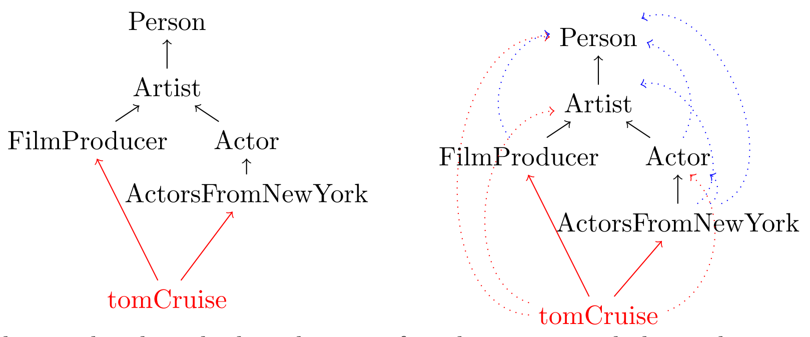
In the following, we introduce some terms and notions that we use. *Materialization* (closure) refers to computing the deductive closure of an ontology (alternatively, making all implicitly stored data explicit by using inference) [15]. *Ontology completion [5]*–refers to computing the closure of the ontology and adding additional instances by following class inclusions in the TBox (for instance, if Actor is a subclass of Artist and the instance Tom is an Actor, add another instance who is not an Actor but is an Artist. In this case, if an instance is not known, one can use anonymous resource to identify the unknown instance). *Loss of semantics*–the transformation of an ontology into a formal context results in loss of semantics if the context mixes TBox (schema axioms) and ABox (instance) data and if the concept lattice obtained from the formal context does not maintain the class hierarchy. Before presenting how a $\mathcal{ELI}$ ontology can be transformed into a formal context, we motivate our approach with an example.

3.1 Motivation

Example 1. Consider the following $\mathcal{ELI}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

$$\begin{aligned} \mathcal{T} = \{ & \texttt{ActorsFromNewYork} \sqsubseteq \texttt{Actor}, \texttt{FilmProducer} \sqsubseteq \texttt{Artist}, \\ & \texttt{Actor} \sqsubseteq \texttt{Artist}, \texttt{Artist} \sqsubseteq \texttt{Person} \} \\ \mathcal{A} = \{ & \texttt{tomCruise}^{\mathcal{I}} \in \texttt{ActorsFromNewYork}^{\mathcal{I}} \} \end{aligned}$$

In order to compare graphical representations of DL ontologies and their corresponding concept lattices, we represent $\mathcal{O}$ and its respective materialization $\mathcal{O}'$ as graphs as shown below:

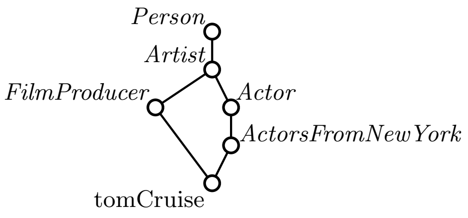


In the graphs, dotted edges denote inferred instance and class subsumption relations.

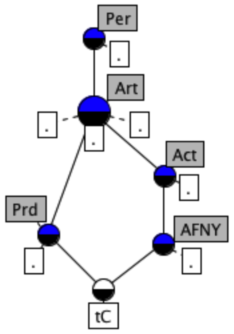
Starting with Example 1, one can ask whether it is possible to obtain a formal context from the ontology $\mathcal{O}$ while maintaining its semantics. The problem here is that DLs and FCA work on different assumptions, i.e, while DL languages are based on the *open world assumption (OWA)*, FCA relies on the *closed world assumption (CWA)*. The former permits to specify only known data whereas the later demands all data should be explicitly specified. To slightly close the gap between these two worlds:

- one can generate the formal context from the closure of the ontology. However, this approach fails when it is not possible to compute the closure of the ontology as this is the case for ontologies created from a DL language equipped with negation and disjunction constructs[5], and
- before transforming the ontology into a formal context, complete the ontology. A drawback of the second approach is that it adds unnecessary data, consequently, giving unwanted results when querying.

[5] `http://www.w3.org/TR/owl2-primer/`



(a) Target lattice associated with the ontology in Example 1.



(b) Lattice associated with the context in Example 2.

Fig. 1: tomCruise (tC) and (.) are objects and the rest are attributes.

To this end, our main objective is to come up with an approach which transforms an ontology into a formal context while maintaining the semantics. Accordingly, a formal context corresponding to the ontology in Example 1 has an associated lattice that looks like the one in Figure 1a. From this onwards, when we speak of this lattice, we refer to it as the *target* lattice. The target lattice *maintains the semantics* because: the class hierarchy of the ontology (TBox) is the same as that of the lattice, and the instance (ABox) and schema (TBox) part of the ontology are treated separately as discussed in Section 3.2.

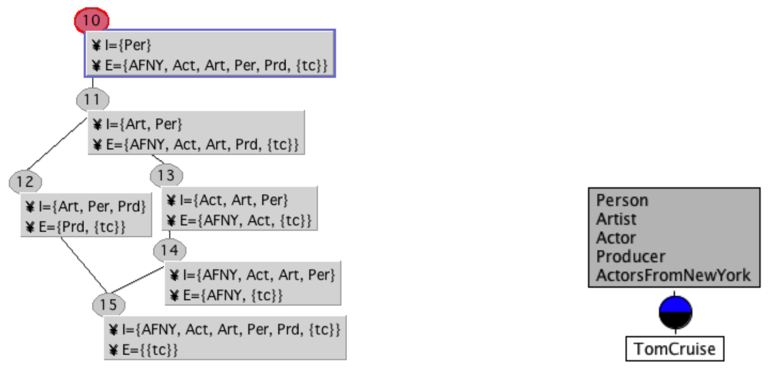
In the following, we provide various formal contexts associated with the ontology in Example 1. For the sake of readability, we shorten concept and individual names as: ActorsFromNewYork (AFNY), FilmProducer (Prd), Actor (Act), Artist (Art), Person (Per), and tomCruise (tC). Consider the following transformations:

Naive approach: materialized ABox

	AFNY	Prd	Act	Art	Per
tC	x	x	x	x	x

This formal context is obtained from the materialized ABox of the ontology. It does not include subclass relations as they can be acquired using *attribute exploration* [9]. But unfortunately, the resulting lattice considers all the attributes to be equivalent, implying loss of semantics, as it can be seen from the lattice in Figure 2b.

Direct approach: materialized ABox and TBox



(a) (b)

Fig. 2: Concept lattice associated with the ontology in Example 1.

	{tC}	AFNY	Prd	Act	Art	Per
{tC}	x	x	x	x	x	x
AFNY		x		x	x	x
Prd			x		x	x
Act				x	x	x
Art					x	x
Per						x

This formal context is produced by taking all the subclass hierarchy of all atomic concepts C and all nominal concepts $\{a\}$ for all individuals a. Formally, a formal context is constructed using: (i) $a^{\mathcal{I}} \in C^{\mathcal{I}}$ into $\{a\}, C \in G, M$, and $(\{a\},\{a\}),(C,C),(\{a\},C) \in I$, and (ii) $C \sqsubseteq D$ into $C, D \in G, M$, and $(C,D),(C,C),(D,D) \in I$. The context is a transformation of the materialized ontology (both the closures of the ABox and TBox are computed as depicted in the right-hand graph of Example 1). The concept lattice of this formal context is shown in Figure 2a. As it can be seen, it does not maintain the semantics because the concept hierarchy is different from that of our target lattice (in Figure 1a). In other words, the concept hierarchy of the concept lattice is different from that of the ontology (in Example 1). Everything is mixed: attributes are also objects and vice versa. Obviously, it is possible to find several other ways of transforming an ontology into a formal context. To avoid any semantic loss, we propose an another approach, where we separately manage the transformation of ABox and TBox assertions. In FCA, attribute exploration is used to discover implicit knowledge. In that, given a concept lattice, a domain expert is asked a series of questions to produce implications that correspond to DL like inclusion axioms. Ontologies contain instance and schema data, where the latter is similar to implications of concept lattices. Hence, when transforming, individuals in the ABox to become objects and concept names to become attributes, besides, assertions in the ABox are transformed into relations. Additionally, class inclusions of the TBox become background implications. The overall transformation procedure leads to a formal context with respect to existing knowledge (this is also known as *background implications* according to [8]). This procedure is formally described in definition 2.

3.2 Proposal

To transform a $\mathcal{ELI}$ knowledge base KB $= \langle \mathcal{T}, \mathcal{A} \rangle$ into a formal context $K = (G, M, I)$, the schema axioms in the TBox become background implications $\mathcal{L}((G, M, I))$ and the ABox assertions become objects, attributes and relations. To elaborate, in K, individuals in the ABox constitute objects G, class names in the ABox and TBox yield attributes in M, and ABox assertions create relations between objects and attributes $I \subseteq G \times M$. Here, we consider acyclic TBoxes so as to avoid class names becoming objects in a context.

Definition 2 (Transforming $\mathcal{ELI}$ Ontologies into Formal Contexts). *We define the transformation of* KB $= \langle \mathcal{T}, \mathcal{A} \rangle$ *into a formal context* (G, M, I) *thanks to a transformation function* σ *as follows:*

– *An axiom* $C \sqsubseteq D$ *in* $\mathcal{T}$ *corresponds to an implication in* $\mathcal{L}((G, M, I))$*, i.e., the set of implications based on* (G, M, I)*:* $C \sqsubseteq D \longmapsto C \rightarrow D \in \mathcal{L}((G, M, I))$.
– *Concept expressions* C *(class name),* $\exists R.C$*, and* $\exists R^-.C$*, correspond respectively to attributes* C*,* $\exists R.C$*, and* $\exists R^-.C$ *in* M.
– *An individual* a *in* $\mathcal{A}$ *corresponds to an object* a *in* G.
– *When* a *is an instance of* C *resp.* $\exists R.C$*,* $\exists R^-.C$*, then* $(a, C) \in I$ *resp.* $(a, \exists R.C) \in I$*,* $(a, \exists R^-.C) \in I$.
– *When* a *is related to* b *through* R*, then* $(a, \exists R.\top) \in I$ *and* $(b, \exists R^-.\top) \in I$.

Example 2. The translation of the ontology in Example 1 into a formal context K and its background implications $\mathcal{L}$ are shown below:

K	AFNY	Prd	Act	Art	Per
tC	x				

$$\mathcal{L} = \{\ \text{AFNY} \rightarrow \text{Act},\ \text{Prd} \rightarrow \text{Art},\ \text{Act} \rightarrow \text{Art},\ \text{Art} \rightarrow \text{Per}\ \}$$

Construction of concept lattices: there are several algorithms that can compute concept lattices associated with a formal context. Some of these are discussed in the literature [9,6] and have also been implemented. They work on an empty implication base. Thus, most are not suitable for contexts with background implications. In [8], the author provides an algorithm for attribute exploration with background implications. This technique can be employed for our purpose. As a result, the concept lattice associated with the formal context and background implications of Example 2 is depicted in Figure 1b.

Next we show that concept lattices associated with $\mathcal{ELI}$ ontologies can be queried by LQL – lattice query language.

4 Querying Concept Lattice

SPARQL query answering over $\mathcal{ELI}$ ontologies can be considered as lattice query answering over $K_{\mathcal{ELI}}$ concept lattices. To do this, we need to introduce a query language for concept lattices. Each node in a lattice can be seen as a query formed by a conjunction of: a concept intent and a concept extent. Intuitively, querying concept lattices amounts to fetching the objects given a set of attributes as query constants, alternatively, fetching the attributes given a set of objects as query constants or terms. Query terms can be connected using the logical operators: AND and OR to form a complex term. A term is either a set of objects called *object term* (OT) or a set of attributes called *attribute term* (AT).

Definition 3 (Object and Attribute Terms). *Given a formal context $K = (G, M, I)$, an object term (OT) and an attribute term (AT) are defined inductively as:*

$$\mathrm{OT} = \{g\} \mid \mathrm{OT}_1 \text{ AND } \mathrm{OT}_2 \mid \mathrm{OT}_1 \text{ OR } \mathrm{OT}_2, \textit{ where } g \in G$$
$$\mathrm{AT} = \{m\} \mid \mathrm{AT}_1 \text{ AND } \mathrm{AT}_2 \mid \mathrm{AT}_1 \text{ OR } \mathrm{AT}_2, \textit{ where } m \in M$$

The expression OT_1 AND OT_2 denotes the greatest lower bound (GLB) in the concept lattice $\underline{\mathcal{B}}(G, M, I)$. The expression OT_1 OR OT_2 denotes the least upper bound (LUB) in $\underline{\mathcal{B}}(G, M, I)$. Dually, the expression AT_1 AND AT_2 denotes the GLB in the concept lattice $\underline{\mathcal{B}}(G, M, I)$ (keeping the orientation of $\underline{\mathcal{B}}(G, M, I)$ based on the extents). Finally, the expression AT_1 OR AT_2 denotes the LUB in $\underline{\mathcal{B}}(G, M, I)$.

Based on the definitions of object and attribute terms, we introduce LQL queries. In this paper, we do not address the problem of negation in the query (and thus set difference).

Definition 4 (LQL - Lattice Query Language). *Given an object term* OT*, an attribute term* AT*, and variables $x, y \in V$ where V is a finite set of variables, an LQL query can take the following forms:*

$$q(y) = (\mathrm{OT}, y);\ q(x) = (x, \mathrm{AT});\ q() = (OT, AT)$$

$q(y), q(x)$, and $q()$ do not necessarily correspond to formal concepts, only when OT and AT are closed sets. If OT is a closed set in $q(y) = (\mathrm{OT}, y)$, then y corresponds to the intent associated with OT. The same thing happens with x when AT is a closed set in $q(x) = (x, \mathrm{AT})$. For evaluating x and y in every possible case we do the following:

- if $\mathrm{OT} = \{g\}$, then $y = \{g\}'$, i.e., all attributes that are associated with the object g.
- if $\mathrm{OT} = \{g_1\}$ AND $\{g_2\}$, then $y = \{g_1, g_2\}'$
- if $\mathrm{OT} = \{g_1\}$ OR $\{g_2\}$, then $y = \{g_1\}' \cup \{g_2\}'$

Similarly, the evaluation of $q(x) = (x, \mathrm{AT})$ is given as follows:

– if AT $= \{m\}$, then $x = \{m\}'$, i.e., all objects that are associated with the attribute m.
– if AT $= \{m_1\}$ AND $\{m_2\}$, then $x = \{m_1, m_2\}'$
– if AT $= \{m_1\}$ OR $\{m_2\}$, then $x = \{m_1\}' \cup \{m_2\}'$

Finally, the evaluation of $q() = (\mathrm{OT}, \mathrm{AT})$ is:

– *true* if OT = AT$'$ or AT = OT$'$ and *false* otherwise.

Example 3. Let us consider querying the concept lattice shown in Figure 3.

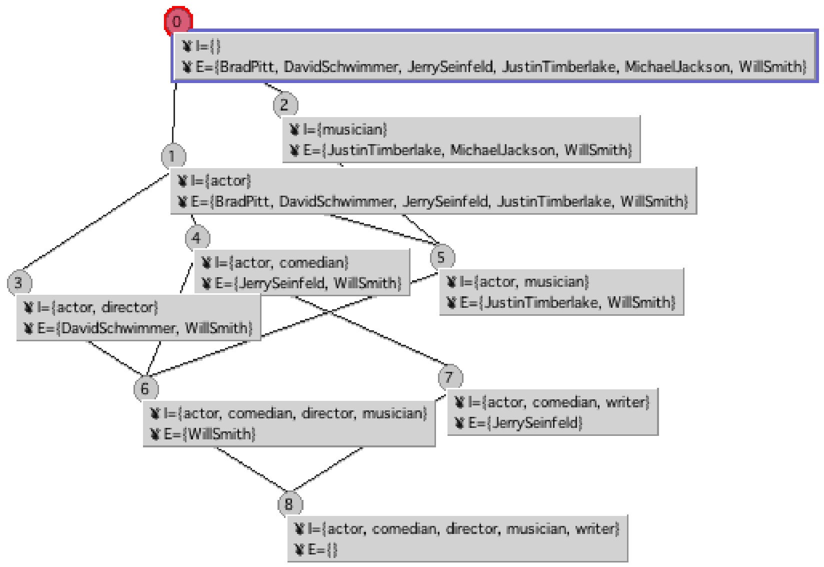

0
I={}
E={BradPitt, DavidSchwimmer, JerrySeinfeld, JustinTimberlake, MichaelJackson, WillSmith}
2
I={musician}
E={JustinTimberlake, MichaelJackson, WillSmith}
1
I={actor}
E={BradPitt, DavidSchwimmer, JerrySeinfeld, JustinTimberlake, WillSmith}
4
I={actor, comedian}
E={JerrySeinfeld, WillSmith}
5
I={actor, musician}
E={JustinTimberlake, WillSmith}
3
I={actor, director}
E={DavidSchwimmer, WillSmith}
7
I={actor, comedian, writer}
E={JerrySeinfeld}
6
I={actor, comedian, director, musician}
E={WillSmith}
8
I={actor, comedian, director, musician, writer}
E={}

Fig. 3: A concept lattice representing artists professions.

– For $q_1(x) = (x, \{actor\}$ AND $\{comedian\}$ AND $\{writer\})$, we have $x = \{JerrySeinfeld\}$.
– $q_2(x) = (x, \{writer\}$ OR $\{director\})$, we have $x = \{DavidSchwimmer, WillSmith, JerrySeinfeld\}$.
– $q_3(y) = (\{JustinTimberlake\}$ AND $\{WillSmith\}, y)$, we have $y = \{actor, musician\}$.
– $q_4(y) = (\{DavidSchwimmer\}$ OR $\{JustinTimberlake\}, y)$, we have $y = \{actor, director, musician\}$.

The complexity of answering LQL queries is *polynomial* in the size of the formal context, i.e., $\mathcal{O}|(G, M, I)|$. The advantage of LQL over SPARQL is that, it allows to compute the least upper bound and greatest lower bound of query answers. We now present one important part of this work which is reducing SPARQL query answering over $\mathcal{ELI}$ ontologies into LQL query answering over $K_{\mathcal{ELI}}$ concept lattices.

5 SPARQL query answering over ontologies vs LQL query answering over concept lattices

Recently, SPARQL has been extended with different entailment regimes and regular path expressions[6]. The semantics of SPARQL relies on the definition of basic graph pattern matching that is built on top of simple entailment [10]. However, it may be desirable to use SPARQL to query triples entailed from subclass, subproperty, range, domain, and other relations which can be represented using DL schema languages such as $\mathcal{ELI}$. The SPARQL specification defines the results of queries based on simple entailment. The specification also presents a general parametrized definition of graph pattern matching that can be expanded to other entailments beyond simple entailment. Query answering under an entailment regime can be achieved via: (1) materialization (computing the deductive closure of the queried graph), (2) rewriting the queries using the schema, and (3) hybrid (combining materialization and query rewriting) [10].

Example 4. Let us consider the evaluation of the SPARQL query Q on the ontology $\mathcal{O}$ and $\mathcal{O}'$ of Example 1. Q = select all those who are artists.

```
SELECT  ?x WHERE {?x a Artist.}
```

Under simple entailment evaluation of a SPARQL query, the answers of Q over $\mathcal{O}$ is empty, i.e., $Q(\mathcal{O}) = \emptyset$. For the reason that, simple entailment is based on graph matching which requires the variable $?x$ in the query to be bound with a term in the graph. Since there is no term where it can be bound to, the result is empty. However, under higher entailment regimes (such as the RDFS entailment [10]) the result of Q is non-empty because inferred instances obtained through reasoning are taken into account for computing the answers. To get a non-empty answers for the above query, one can use one of the following approaches:

1. *Materialization:* involves all implicit data to be computed before the evaluation of the query. This can be done by using a DL reasoner. Consequently, in Example 1, the materialization of $\mathcal{O}$ is $\mathcal{O}'$. Thus, the evaluation of Q over $\mathcal{O}$ is $Q'(\mathcal{O}) = \{\text{tomCruise}\}$.
2. *Query rewriting:* is the task of converting a SPARQL query into one that involves schema axioms. It can be done using SPARQL property paths (a.k.a. regular path expressions). For instance, the above query can be rewritten as:

   ```
   SELECT  ?x WHERE {?x a/⊑* Artist.}
   ```

   This query Q' selects all instances of Artist and that of its subclasses by navigating through the subclass relation ($\sqsubseteq^*$). The rewriting can be evaluated over $\mathcal{O}$ to obtain $Q'(\mathcal{O}) = \{\text{tomCruise}\}$.

In summary, materialization requires a reasoner to expand the knowledge base, the complexity of this task depends on the type of the schema language. On the other hand, query rewriting requires modifying query patterns using SPARQL

[6] http://www.w3.org/TR/sparql11-query/

property paths. This also results in a further jump in the complexity of query answering.

As described above, unlike SPARQL query answering over ontologies, query answering over a concept lattice is relatively easier. Due to the fact that once the concept lattice is obtained from the ontology, LQL can be used to query the lattice. Consequently, alleviating those expensive tasks. The above SPARQL query can be converted into an LQL query as: $q(x) = (x, Artist)$. The evaluation of this query over a concept lattice obtained from $\mathcal{O}$ is as expected $Q'(\mathcal{O}) = \{\text{tomCruise}\}$.

The complexity of SPARQL query answering over $\mathcal{ELI}$ ontologies is larger than that of LQL query answering over $K_{\mathcal{ELI}}$ concept lattices. Since, the expressive power of SPARQL is superior than that of LQL. For $\mathcal{ELI}$ ontologies a query language like LQL is sufficient to retrieve individuals (or objects) and classes (or attributes).

6 Related work

To date, several studies have been carried out to assess the relevance and benefits of FCA for DL [5,3,4,14,7,12]. Notably, the work in [14] presents a survey on the advantageous of FCA for DL ontologies. Accordingly, some of the benefits that FCA can bring to the DL world include: knowledge discovery, extended subsumption hierarchy (of conjunctions of concepts) [1], subsumption hierarchy of least common subsumers, exploring finite models [3,4], role assertion analysis, supporting bottom-up construction and completion of ontologies. Since the survey, other studies, [7] and [12], have carried out experiments to characterize and analyse SW data using FCA tools. The former provides an entry point to a linked data using questions in a way that can be navigated. It gives a translation of an RDF graph into a formal context where the subject of an RDF triple becomes the object, a composition of the predicate and object of the triple becomes an attribute. The latter obliges the user to specify objects and attributes of a context. With that, it creates SPARQL queries to extract content from linked data in order to populate the formal context. Despite the fact that all these works have employed FCA techniques, to the best of our knowledge, none of them provide a formal and precise translation of ontologies into a formal context as we did here.

7 Conclusion

In this work, firstly, we have proposed a formal transformation of $\mathcal{ELI}$ ontologies into formal contexts. This enables to benefit from some advantages that FCA may offer to the DL world. Then we have shown that SPARQL query answering over $\mathcal{ELI}$ ontologies can be considered as lattice query answering over $K_{\mathcal{ELI}}$ concept lattices. This alleviates some reasoning and query rewriting tasks that are required for SPARQL query answering.

Moreover, even if there already exist substantial work relating DL, semantic web and FCA, there remains a lot of research work to be carried out. Such a research work is concerned with the correspondence between concept lattices from FCA and DL-based class hierarchies, query answering and information retrieval, and scalability as well. In addition, as FCA could benefit from DL-based reasoning capabilities, semantic web and DL-driven applications can take advantage of FCA-based ontology design, data analysis and knowledge discovery capabilities of FCA.

In the future, we plan to extend and experiment with the proposed approach. We will investigate how well it scales, given the size of ontologies.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2007), iSBN 9780511717383
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: IJCAI. vol. 5, pp. 364–369 (2005)
3. Baader, F., Distel, F.: A finite basis for the set of el-implications holding in a finite model. In: ICFCA. pp. 46–61. Springer (2008)
4. Baader, F., Distel, F.: Exploring finite models in the description logic EL gfp. In: ICFCA. pp. 146–161. Springer (2009)
5. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Proc. of IJCAI. vol. 7, pp. 230–235 (2007)
6. Carpineto, C., Romano, G.: Concept data analysis: Theory and applications. Wiley (2004)
7. d'Aquin, M., Motta, E.: Extracting relevant questions to an RDF dataset using formal concept analysis. In: Proceedings of the sixth international conference on Knowledge capture. pp. 121–128. ACM (2011)
8. Ganter, B.: Attribute exploration with background knowledge. Theoretical Computer Science 217(2), 215 – 233 (1999)
9. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Berlin (1999)
10. Glimm, B.: Using SPARQL with RDFS and OWL entailment. Reasoning Web. Semantic Technologies for the Web of Data pp. 137–201 (2011)
11. Hayes, P.: RDF semantics. W3C Recommendation (2004)
12. Kirchberg, M., Leonardi, E., Tan, Y.S., Link, S., Ko, R.K., Lee, B.S.: Formal concept discovery in semantic web data. In: ICFCA. pp. 164–179. Springer-Verlag (2012)
13. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Rec. (2008)
14. Sertkaya, B.: A survey on how description logic ontologies benefit from FCA. In: CLA. vol. 672, pp. 2–21 (2010)
15. Ter Horst, H.: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3), 79–115 (2005)