Mining of $\mathcal{EL}$-GCIs

Daniel Borchmann and Felix Distel
Faculty of Computer Science
TU Dresden
Dresden, Germany
{borch,felix}@tcs.inf.tu-dresden.de

***Abstract*—We consider an existing approach for mining general inclusion axioms written in a lightweight Description Logic. In comparison to classical association rule mining, this approach allows more complex patterns to be obtained. Ours is the first implementation of these algorithms for learning Description Logic axioms. We use our implementation for a case study on two real world datasets. We discuss the outcome and examine what further research will be needed for this approach to be applied in a practical setting.**

***Index Terms*—Description Logics, General Inclusion Axioms**

I. Introduction

Data mining has traditionally focused on the extraction of relatively simple, rigid patterns from datasets. For example, in association rule mining the patterns are simple pairs of itemsets. Their semantics is that transactions that contain the first itemset also contain the second. In this paper we implement and test an approach for mining a more complex and flexible type of associations, namely general concept inclusions (GCIs) that are written in the Description Logic (DL) $\mathcal{EL}$ [1]. Our hope is that this combination of data mining and DL will be mutually beneficial for both fields.

Classical association rule mining requires a simple type of data where individuals (or transactions) have attributes (or items) associated with them [2]. In many data sets there are also binary relations between individuals, e.g. there might be a relation linking two transactions if they have been performed by the same customer. In another scenario, the data may have been obtained from a social network and the binary relations are the various degrees of friendship among individuals. Classical association rules cannot take these relations into account, at least not natively. Data mining (and in particular social graph mining) might benefit from the use of a language (e.g. a language from the DL family) that is inherently capable of talking about binary relations.

Conversely, DL might benefit from a data mining technique that can assist in the design of ontologies. Ontology design can be a tedious and error-prone task. It is usually done by domain experts, who are not experts in logics. By mining GCIs from existing data one could obtain a starting point for building an ontology. Ideally, the results of the mining process should be verified by a domain expert.

The work in this paper is based on an algorithm presented in [3], [4]. Ours is the first implementation of this algorithm. The main purpose of this work is to serve as a prove of concept that mining GCIs can yield useful results.

Unlike algorithms for mining association rules, which usually try to compute all possible association rules (with sufficient support and confidence), this algorithm only computes a compact representation of the GCIs, a so-called *base*. This is necessary since even for a finite number of relations and attributes there are infinitely many possible $\mathcal{EL}$-GCIs.

We have chosen two datasets with different characteristics for testing our implementation. The first is an excerpt from DBpedia [5], namely those individuals who either have children or are someone's child. As this dataset has been semi-automatically extracted from Wikipedia, it contains relatively many errors. Thus, it can teach us valuable lessons on how the algorithm behaves on noisy and incomplete data. Second, we have used the DrugBank [6], which is part of the Linked Open Data Cloud [7]. It contains data about various drugs and their protein targets. In comparison to DBpedia it is relatively free of errors. We test the algorithm on both datasets and discuss the outcome. The results reveal potential, but they also give indications in what areas more research will be needed (cf. Section V).

The structure of the paper is divided as follows. We start with a short introduction to $\mathcal{EL}$ and the mining algorithm. This is followed by a description of our implementation and the two datasets. We present some of our observations and finally discuss what lessons can be learned from them.

II. Theoretical Foundations

A. The Description Logic $\mathcal{EL}$

$\mathcal{EL}$ is a lightweight Description Logic. Although being relatively inexpressive compared to other DLs it has gained widespread acceptance, e.g. in biomedical ontologies [8], [9] and within the latest OWL standard that contains the profile OWL 2 EL [10]. $\mathcal{EL}$'s success is largely due to its favorable algorithmic properties.

We give an overview over the syntax of $\mathcal{EL}$ as well as an intuition for its semantics to the extent which is needed to understand this work. For a more formal introduction to DL consider [1]. Like all Description Logics $\mathcal{EL}$ can be used to describe individuals and classes of individuals, using *concept descriptions*. The main building blocks for concept descriptions are a set $\mathcal{N}_C$ of *concept names* and a set $\mathcal{N}_R$ of *role names*. Concept names, e.g. Parent, Child or FictionalCharacter, are themselves concept descriptions. Role names describe relationships between individuals such as hasChild, createdBy, etc. New concept descriptions can be

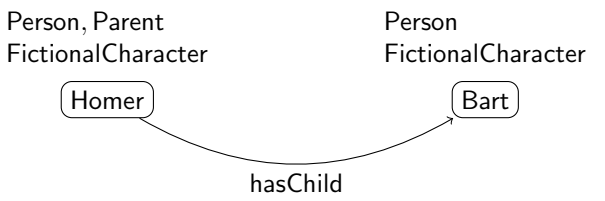

Fig. 1: A Simple Model

constructed using the constructors *top* ($\top$), *conjunction* ($\sqcap$) and *existential restrictions* ($\exists$): If C and D are concept descriptions and r is a role name, then $C \sqcap D$ and $\exists r.C$ are also concept descriptions. For example, the concept description

$$\mathsf{Parent} \sqcap \mathsf{FictionalCharacter} \tag{1}$$

describes individuals that are both parents and fictional characters. The description

$$\exists \mathsf{hasChild}.\mathsf{Parent} \tag{2}$$

describes individuals with children who are themselves parents. More formally, the semantics of $\mathcal{EL}$ is based on *models* $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$. A model consists of a set $\Delta_\mathcal{I}$ called *domain* and a function $\cdot^\mathcal{I}$ mapping each concept name A to a subset $A^\mathcal{I} \subseteq \Delta_\mathcal{I}$ and each role name r to a relation $r^\mathcal{I} \subseteq \Delta_\mathcal{I} \times \Delta_\mathcal{I}$. The function $\cdot^\mathcal{I}$ can be extended recursively to map every concept description C to its interpretation $C^\mathcal{I}$ by defining $\top^\mathcal{I} = \Delta_\mathcal{I}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$ and $(\exists r.C)^\mathcal{I} = \{x \in \Delta_\mathcal{I} \mid \exists y \in C^\mathcal{I} \colon (x, y) \in r^\mathcal{I}\}$. Each *model* can be represented by a directed graph, the so-called *description graph* of $\mathcal{I}$, whose edges are labeled with a role name and whose nodes are labeled with sets of concept names as in Figure 1. In Figure 1 the description (1) is interpreted as $\{\mathsf{Homer}\}$ and, as there are no grandparents in this model, the description (2) is interpreted as the empty set.

Relationships between concepts can be expressed in the form of general concept inclusions (GCIs). These are statements of the form $C \sqsubseteq D$ where C and D are concept descriptions. Consider for example

$$\mathsf{Person} \sqcap \exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Parent}, \tag{3}$$

which expresses that every person who has a child is a parent. A GCI *holds in a model* if the interpretation of its left hand side is contained in the interpretation of its right hand side, e.g. the GCI from (3) holds in the model from Figure 1, because both sides have the interpretation $\{\mathsf{Homer}\}$. A concept description C is *more specific* than a concept description D if the GCI $C \sqsubseteq D$ holds in all possible models. If $\mathcal{B}$ is a set of GCIs and $C \sqsubseteq D$ is another GCI, then we say that $C \sqsubseteq D$ *follows semantically* from $\mathcal{B}$ if and only if in every model $\mathcal{I}$ in which all GCIs from $\mathcal{B}$ hold, the GCI $C \sqsubseteq D$ holds as well.

Observe that ordinary association rules can be expressed as a simple form of GCIs where only conjunction and no existential restrictions are used. An association rule stating that people who buy cereals and honey also buy milk could be expressed as $\mathsf{BuyerOfCereal} \sqcap \mathsf{BuyerOfHoney} \sqsubseteq \mathsf{BuyerOfMilk}$. $\mathcal{EL}$-GCIs are more expressive whenever relations exist between individuals. In such a situation an infinite number of concepts can be described using $\mathcal{EL}$ (Usually, only a fraction of them will be interesting in practice). Therefore, we have tested our approach only on datasets where relations between individuals exist. The two datasets used in this work are presented in Section III.

B. The Basic Mining Algorithm

We provide an overview of the theory that is involved. It is described in detail in [3], [4]. In conventional association rule mining there is only a finite number of items and therefore only a finite number of rules. When using $\mathcal{EL}$-GCIs to describe associations this is no longer the case, as it allows for nesting of existential restrictions, causing a blowup. Even worse, the number of possible concept descriptions grows faster than exponential with the number of nestings (the so-called *role depth*). Hence, even with a restricted role depth we cannot expect to enumerate all associations that hold in the data in reasonable time. Instead, we enumerate a *base* of the $\mathcal{EL}$-GCIs holding in the data, i.e. a set $\mathcal{B}$ of GCIs such that

- all GCIs from $\mathcal{B}$ hold in the model, and
- all GCIs that hold in the model follow semantically from $\mathcal{B}$.

In [3], [4] a first algorithm for computing a finite base from a given dataset is presented. Its main purpose is knowledge base completion, and as a knowledge base completion algorithm it competes with other methods, e.g. [11], [12]. Here we employ it in a slightly different setting, where no knowledge base, but a large dataset is initially present.

The algorithm proceeds in two steps. First, since it is impossible to consider all possible $\mathcal{EL}$-concept descriptions, a finite set of relevant concept descriptions is obtained from the data using a technique called model-based most specific concepts. For a given model $\mathcal{I}$ and a set $X \subseteq \Delta_\mathcal{I}$ the concept description C is called the *model-based most specific concept* of X if it is the most specific concept description that still satisfies $X \subseteq C^\mathcal{I}$. The model-based most specific concept of X is denoted by $X^\mathcal{I}$. The set of relevant concept descriptions is then obtained as

$$M_\mathcal{I} = \mathcal{N}_C \cup \{\exists r.X^\mathcal{I} \mid X \subseteq \Delta_\mathcal{I}\}.$$

In [4] it is shown that the GCIs where both sides are conjunctions of these relevant concept descriptions already form a base $\mathcal{B}$. This base usually contains a large number of redundancies.

Lemma 1. *The set of GCIs*

$$\mathcal{B} = \{C \sqsubseteq (C^\mathcal{I})^\mathcal{I} \mid C \in R_\mathcal{I}\}$$

where $R_\mathcal{I} = \{\bigsqcap U \mid U \subseteq M_\mathcal{I}\}$ *is a base for the model* $\mathcal{I}$.

Whenever only conjunctions occur in a GCI its semantics become simpler. This allows the use of established theories to further reduce the size of $\mathcal{B}$. In the second step of the algorithm this is done using a mathematical theory called Formal Concept Analysis (FCA) [13]. FCA uses a data structure

called *formal contexts*. A formal context $\mathbb{K}$ is defined to be a triple $\mathbb{K} = (G, M, I)$ where G is a set of objects, M is a set of attributes and $I \subseteq G \times M$ is a binary relation. *Pseudo-intents* are subsets of M with certain special properties and that can be used to characterize the implicational theory of a formal context (cf. [13] for a formal definition). Using an algorithm from FCA, called *Next-Closure*, pseudo-intents can be computed. A smaller base $\mathcal{B}'$ can be obtained using the following result from [4].

Lemma 2. *Let $\mathcal{I}$ be a model and $\mathbb{K}_\mathcal{I} = (\Delta_\mathcal{I}, M_\mathcal{I}, I)$ be the context where I is defined to contain the pair (x, C) iff $x \in C^\mathcal{I}$. The set of GCIs*

$$\mathcal{B}' = \left\{ \bigsqcap P \sqsubseteq ((\bigsqcap P)^\mathcal{I})^\mathcal{I} \mid P \text{ pseudo-intent of } \mathbb{K}_\mathcal{I} \right\}$$

is a base for the model $\mathcal{I}$.

Given $\mathbb{K}_\mathcal{I}$ as input Next-Closure successively returns new GCIs (or more precisely their left hand sides), however the delay between two GCIs can be exponentially large [14]. It has been a longstanding problem in FCA whether improvements to Next-Closure can be made, yet no breakthrough has been achieved so far. It can be shown that $\mathcal{B}'$ has minimal cardinality among all bases of the $\mathcal{EL}$-GCIs holding in a given dataset.

Notice that the algorithm will only compute GCIs that *hold* in a given dataset. In data mining terminology one could say it computes associations with confidence 1. The reasons for this and its implications will be discussed in Section IV-C.

C. Modifications to the Basic Algorithm

The basic algorithm described above consists of two parts, first the relevant concept descriptions are computed and then Next-Closure is used to obtain the basis. During the first step a long delay can occur during which no output is produced. To avoid this a modification to the basic algorithm is suggested in [4] where relevant concept descriptions are computed on the fly, i.e. during the execution of Next-Closure. Our implementation uses this modified approach.

Models that occur in practice can be cyclic, i.e. the graph that represents them is not a tree or forest. In this case the existing strategy for obtaining the relevant concept descriptions does not work when $\mathcal{EL}$ is used. This makes it necessary to use an extension of $\mathcal{EL}$ namely the DL $\mathcal{EL}_{\text{gfp}}$ that allows for cyclic concept descriptions. Using $\mathcal{EL}_{\text{gfp}}$ the cycles in the model can be mirrored within the concept descriptions. In this work, one of the models (DBpedia) contains cycles, while the other does not.

III. Description of Data Used

We have implemented the $\mathcal{EL}$-exploration algorithm prototypically as part of a larger program called `conexp-clj`, which has been developed for Formal Concept Analysis. The implementation language is Clojure, a dialect of Lisp running on the Java Virtual Machine. The implementation itself provides an abstract handling of Description Logics and interpretations and allows for the examination of properties of the algorithm. Not necessarily is it designed for performance. Nevertheless, some optimizations ideas have been implemented and evaluated. The descriptions of these requires more technical insight into the algorithm and is therefore out of the scope of this paper. For details see [15].

We have tested the algorithm on two different data sets of different quality and size. In the next two subsections we describe these two data sets and how they have been constructed from data from the Linked Open Data Cloud [7].

A. Child-Relation in DBpedia

The first data set has been extracted from data from the DBpedia project [5]. The aim of this project is to automatically extract relational data from Wikipedia infoboxes. The data snapshot from March 2011, which has been used for our experiments, contains over 10 million RDF triples. To handle them all at once is out of the scope of our capabilities, yet. Instead, we have chosen a small subset of the data to construct an $\mathcal{EL}$-model for our experiments.

To this purpose, we have used two data sets of RDF triples from the DBpedia, one containing relations between individuals and the other containing instances of certain classes. We then chose the role http://dbpedia.org/ontoloy/child and collected all RDF triples from the relational data set labeled with this relation. The individuals which appeared in these collected triples formed the base set Δ_{DB} for our model. Then the concept names were chosen from the data set containing the instances. Here all those classes have been chosen which had at least one instance in Δ_{DB}. In sum, this procedure resulted in an $\mathcal{EL}$-model $\mathcal{I}_{\text{DB}}$ containing 5626 individuals, one role relation and 60 concept names.

As we shall see later, this model has some odd properties which are due to numerous errors in the DBpedia data sets. First of all, the child relation we have chosen proves to be cyclic in this interpretation. Furthermore, the relation does not exist between persons alone but also between authors and their work or even between persons and places. Those errors are mostly owed to the nature of Wikipedia's infoboxes which are not standardized in any way. Thus, the data collected by the DBpedia project has to be normalized in certain ways and often enough errors occur. We shall see later what consequences this has for our algorithm and the computed results.

B. DrugBank

As another data set from the Linked Open Data Cloud we have chosen the DrugBank [6] and Diseasome [16] data sets. They provide information about drugs, their possible disease targets and information about genes that are influenced by given drugs. Three different roles occur in the datasets, namely *isSubtypeOf* between diseases describing an is-a relation, *treatedBy* from diseases to drugs and *targets* from drugs to protein sequences being affected by the given drug. In contrast to the DBpedia model we construct the concept names for this model by looking at certain RDF triples. More precisely, we collect the RDF triples named with

- http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory
- http://www4.wiwiss.fu-berlin.de/diseasome/resource/diseasome/class
- http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/goClassificationProcess.

Every individual appearing on the left side of one such RDF triple is associated with the right side of the triple as its concept name. The first of the three relations associates drugs with their category in the DrugBank data set, such as being an antianemic agent, an anticoagulant, an antiviral agent and so on. In a similar fashion classes like metabolic, endocrine, immunological are associated with diseases by the second relation.

The third relation associates protein sequences with their processes they are involved in. These processes have a verbatim description in the corresponding RDF triples and are therefore quite many, resulting in a lot of different concept names if taken literally. We therefore did a very simple clustering by searching for words like "metabolism", "signal transduction" or "homeostasis" in those process descriptions and classified the genes according to the first match found. Undoubtedly, this is not a very sophisticated method of clustering the processes, but it is an easy method to keep the example small and controllable.

The construction described so far results in an $\mathcal{EL}$-model $\mathcal{I}_{\text{drugs}}$ with 13335 individuals, three role relations and 899 concept names. In contrast to the model constructed from the DBpedia data set, the model $\mathcal{I}_{\text{drugs}}$ is supposed to be of much higher quality. Furthermore, the roles have been chosen such that the resulting model $\mathcal{I}_{\text{drugs}}$ is acyclic.

IV. Results

A. Results for DBpedia

We have tested our implementation on the model derived from the child-relation from DBpedia as described in Section III-A. In total, 1252 GCIs were found. Among these only the first 339, i.e. less than one third, are of a form that could have been obtained without using a Description Logic. More precisely, they were either of the form

$$\prod_{C\in A} C \sqsubseteq \prod_{C\in B} C, \tag{4}$$

or

$$\prod_{C\in A} C \sqsubseteq \mathsf{All}. \tag{5}$$

for some sets of concept names A and B. All stands for the concept description that combines all possible attributes that an individual can have. Intuitively, (5) states that an individual that belongs to all concepts from A must belong to any $\mathcal{EL}_{\text{gfp}}$-concept. In practice, this usually occurs when there is no individual belonging to all concepts from A in the model.

Examples for obtained GCIs are the following

$$\begin{aligned} \mathsf{Chancellor} &\sqsubseteq \mathsf{Politician} \\ \mathsf{Book} &\sqsubseteq \mathsf{Work} \\ \mathsf{Person} \sqcap \mathsf{Place} &\sqsubseteq \mathsf{All} \\ \mathsf{Astronaut} \sqcap \mathsf{Model} &\sqsubseteq \mathsf{All} \end{aligned} \tag{6}$$

The second and third GCI in (6) refer to concepts such as Book or Place that one would not expect to apply to persons. Since the model that we use is restricted to individuals occurring in the child-relation, one would expect all individuals to be persons. Book, Work, and Place still occur because of errors in DBpedia. These errors result from the process in which DBpedia is created. We discuss ways to address flawed data in Section IV-C1.

The algorithm has a tendency to enumerate GCIs in an order of increasing role depth of the left hand sides. More details on why this is the case can be found in [3]. This is a desired property, since in our tests GCIs with very large role depth were more likely to be too specific due to overfitting and therefore less interesting.

For the DBpedia model the GCIs of role depth 1 were mostly of the form $A \sqcap \exists \mathsf{child}.\top \sqsubseteq \mathsf{All}$. One example is

$$\mathsf{Model} \sqcap \exists \mathsf{child}.\top \sqsubseteq \mathsf{All},$$

which states that individuals belonging to Model that have a child successor must belong to all possible concepts. More bluntly speaking, no offspring of a fashion model has ever gained enough relevance to merit a DBpedia entry.

The GCIs appearing later are more complex. For example, the relatively general GCI

$$\exists \mathsf{child}.\mathsf{Person} \sqsubseteq \mathsf{Person} \tag{7}$$

is found (which applies to 1359 individuals), but also very specific GCIs with larger role depths appear, such as

$$\begin{aligned} \exists \mathsf{child}.\mathsf{Artist} \sqcap \exists \mathsf{child}.\mathsf{Politician} \sqcap \mathsf{Person} \sqsubseteq \\ \sqsubseteq \exists \mathsf{child}.\mathsf{Actor} \sqcap \exists \mathsf{child}.\mathsf{OfficeHolder} \\ \sqcap \exists \mathsf{child}.\mathsf{Congressman} \sqcap \mathsf{OfficeHolder}, \end{aligned} \tag{8}$$

which is only applicable to the individual Robert_F._Kennedy. Notice that, while (7) is most likely a desired outcome, (8) is an artefact of incomplete data. This problem shall be discussed Section IV-C1. Towards the end of the exploration, the GCIs become even more specific and much more convoluted, resulting in expressions which are hardly understandable.

B. Results for DrugBank

Owing to the mining algorithm's tendency to produce GCIs of lower role depth earlier, the first GCIs are of relatively simple nature. In our DrugBank model there are concepts that only apply to drugs, such as Nootropics for drugs that belong to the class of nootropics, and concepts that only apply to protein targets, such as RNA_processing for proteins that are involved in RNA processing. No individual can be both a drug and a protein target, thus conjunctions of such

concepts must be unsatisfiable. This results in a large number of GCIs of the form $\bigsqcap_{C\in A} C \sqsubseteq \mathsf{All}$. Furthermore, all drugs in the DrugBank have a target, yielding GCIs of the form $\bigsqcap_{C\in A} C \sqsubseteq \exists \mathsf{target}.\top$. In total, about 1700 GCIs of these two forms were found.

Among the more interesting GCIs relatively few were of role depth 0, one example is

$$\mathsf{tRNA\_processing} \sqsubseteq \mathsf{RNA\_processing} \sqcap \mathsf{Metabolism} \sqcap \mathsf{Physiological\_process}.$$

Quite a large number of GCIs of role depth 1 are found, such as

$$\mathsf{MuscarinicAntagonists} \sqsubseteq \exists \mathsf{targets}\big(\mathsf{Signal\_transduction} \sqcap \mathsf{Cell\_communication} \sqcap \mathsf{Cellular\_process}\big)$$

or more complex ones like

$$\begin{aligned}&\mathsf{Anti\text{-}inflammatory\_LocallyApplied} \\ &\quad \sqsubseteq \mathsf{Anti\text{-}allergicAgents} \sqcap \mathsf{AntiulcerAgent\_topical} \\ &\sqcap \exists \mathsf{targets}.\big(\mathsf{Response\_to\_biotic\_stimulus} \sqcap \mathsf{Defense\_response} \\ &\quad \sqcap \mathsf{Immune\_response} \sqcap \mathsf{Response\_to\_stimulus}\big) \\ &\quad \sqcap \exists \mathsf{targets}.\big(\mathsf{Signal\_transduction} \\ &\sqcap \mathsf{G\text{-}protein\_coupled\_receptor\_protein\_signaling\_pathway}\big).\end{aligned}$$

Notice that in the second GCI the two existential restrictions really refer to two distinct targets, one (typically Interleukin-3) is responsible for an Immune_response and another (e.g. Cysteinyl_leukotriene_receptor1) is responsible for Signal_transduction.

Unlike the DBpedia, which is the result of a semiautomated process, the DrugBank has been created by human experts. Hence can be assumed to contain relatively few errors. Therefore, one could imagine that the results of the mining process form a good starting point for the construction of an ontology. Ideally, each of them should nevertheless be verified by an expert.

C. Observations

1) Insufficient or Noisy Data: Two types of shortcomings in the data can affect the results of the mining. First, there may be incomplete data. In such a situation there might be GCIs that one would not expect to hold, but for which there is no counterexample present in the data. Second, noise or factual errors can cause the opposite problem. There may be GCIs that one would expect to hold, but the data contains a (faulty) counterexample. Both of these problems occur in the DBpedia model, while the DrugBank is relatively free of errors in comparison. We present some examples from DBpedia to illustrate these issues.

Insufficient Data: One example for this issue is the GCI (8). This GCI holds in the model since no other person with an artist and a politician as children has a DBpedia entry. Yet, in reality such persons are nevertheless likely to exist.

Errors in the Data: Among the output of the DBpedia-Exploration is the following fairly incomprehensible GCI.

$$\begin{aligned}&\mathsf{Person} \sqcap \exists \mathsf{child}.\Big(\mathsf{Person} \sqcap \exists \mathsf{child}.\big(\mathsf{Person} \sqcap \\ &\quad \exists \mathsf{child}.(\mathsf{Person} \sqcap \exists \mathsf{child}.\exists \mathsf{child}(\mathsf{Person} \sqcap \exists \mathsf{child}.\top))\big)\Big) \\ &\quad \sqsubseteq \exists \mathsf{child}.\Big(\mathsf{Person} \sqcap \exists \mathsf{child}.\big(\mathsf{Person} \sqcap \\ &\exists \mathsf{child}.(\mathsf{Person} \sqcap \exists \mathsf{child}.(\mathsf{Person} \sqcap \exists \mathsf{child}.\exists \mathsf{child}.\mathsf{Person}))\big)\Big)\end{aligned}$$

It roughly states that a fourth great-grandparent's second great-grandchild must be a person. It is clearly true, but unnecessarily complicated. It occurs only because the much simpler GCI

$$\mathsf{Person} \sqcap \exists \mathsf{child}.\top \sqsubseteq \exists \mathsf{child}.\mathsf{Person}, \tag{9}$$

from which it could be concluded, is not among the output. There is a whole range of false counterexamples, that seemingly disprove (9).

The individual named Bertolt_Brecht is one such example. Bertolt_Brecht has two child-successors, Frank_Banholzer and Stefan_Brecht. Both are not instances of Person. It is surprising to see that this is not an exceptional counterexample, as 1188 individuals in our DBpedia-model contradict (9). Some of these reveal even more oddities in the DBpedia, largely due to incorrect use of the child-relation. For example the individual John_Perkins (the author) has April_1982 as one of his children, most likely because his real daughter was born in April 1982. Other examples include places as children and literary works as children of their authors.

Combinations of Errors and Insufficient Data: Among the output is the GCI

$$\begin{aligned}&\mathsf{Person} \sqcap \exists \mathsf{child}.\Big(\mathsf{Person} \sqcap \exists \mathsf{child}.\big(\mathsf{Person} \sqcap \exists \mathsf{child}.(\mathsf{Person} \\ &\sqcap \exists \mathsf{child}.\mathsf{Artist})\big)\Big) \sqcap \exists \mathsf{child}.\Big(\mathsf{Person} \sqcap \exists \mathsf{child}.\big(\mathsf{Person} \sqcap \\ &\quad \exists \mathsf{child}.\exists \mathsf{child}.(\mathsf{Person} \sqcap \exists \mathsf{child}.\top)\big)\Big) \\ &\quad \sqsubseteq \big(C, \{C \equiv \mathsf{Writer} \sqcap \exists \mathsf{child}.C\}\big)\end{aligned}$$

Just like (8) it has a complicated left hand side that is only satisfied by one individual, namely Carol_Ann_Duffy, the UK's poet laureate. Hence it owes its existence to incomplete data. What makes it even more odd is its right hand side. The right hand side is an $\mathcal{EL}_{\mathrm{gfp}}$-concept description describing individuals whose child is a Writer whose child is a Writer, and so on infinitely. This peculiar consequence is due to an error where both Carol_Ann_Duffy and her husband are each labeled as their respective spouse's child. As we can see, the combination of effects from incomplete and false data can lead to very undesirable artifacts.

Approaches for Dealing with Noise and Incompleteness: A supervised approach for dealing with incomplete data has been examined in [3], [4]. Each of the newly found GCIs is presented to an expert who can accept it or reject it by providing a counterexample. This method is quite sensitive

with respect to errors in the data. It is suitable when the data can be assumed to be largely correct. Since expert interaction is usually costly it should in most situations be applied on a limited scale or with restricted role depths. The DrugBank appears to be a good application for this approach, while DBpedia does not.

An approach that we would like to investigate in the future is by adapting the notions of support and confidence to GCIs. For a more detailed discussion of this approach see Section V.

2) The Influence of Roles: One of the distinguishing features of the presented algorithm is the added expressivity provided by a DL language that can talk about roles in addition to concepts. It also causes some new problems, among the problem of cyclic roles. To test the impact of cyclic roles on the overall behavior of the exploration we have extended both our DBpedia model and the DrugBank model with cyclic roles. More precisely, we have constructed a model $\mathcal{I}_{\text{cycDB}}$ from the given DBpedia data sets in the same way as described in III-A but instead of only considering the child relation we also added the relation of having someone as mother or father. The resulting model proved to be very hard to be explored. Instead of the 4.5 hrs needed to explore the original model $\mathcal{I}_{\text{DB}}$, after three days of runtime only 1700 GCIs had been collected and the computation of the next GCI took over half an hour. Furthermore, due to the highly cyclic nature of $\mathcal{I}_{\text{cycDB}}$ most of the resulting GCIs were very hard to understand and thus of very little value.

However, things got even worse when we considered the relation *interactsWith* given in the DrugBank data set, modeling the information which two drugs interact with each other. Obviously, this relation is highly cyclic (but not symmetric, due to some technical reasons). If we add this relation to our model $\mathcal{I}_{\text{drugs}}$ exploration gets unfeasible with our implementation. The reason for this is that two drugs, Bumetanide and Furosemide, show up as two individuals for which a description graph has to be computed which represents the model based most specific concept of both of them. However, both drugs are very similar and have a large number of other drugs they interact with, resulting in a description graph of over 5 million vertexes. Our prototypical implementation was not built to handle such large description graphs and therefore we could not conduct the exploration. However, before this large description graph appeared, 6 GCIs had been collected, one of which contained a non-trivial cyclic concept description as it's conclusion. But this description was too large to be understandable, showing that even when the exploration yields GCIs in this case, the result might be of little value for the domain expert.

3) Time and Space Behavior: The time and space behavior may give insights into properties of the algorithm itself and we want to discuss our corresponding observations in this section. For the exploration of the DBpedia-model they are depicted graphically in Figure 2.

The time behavior of the exploration shows some kind of exponentially growing delay between the computation of two successive GCIs. Since the FCA part of the implementation

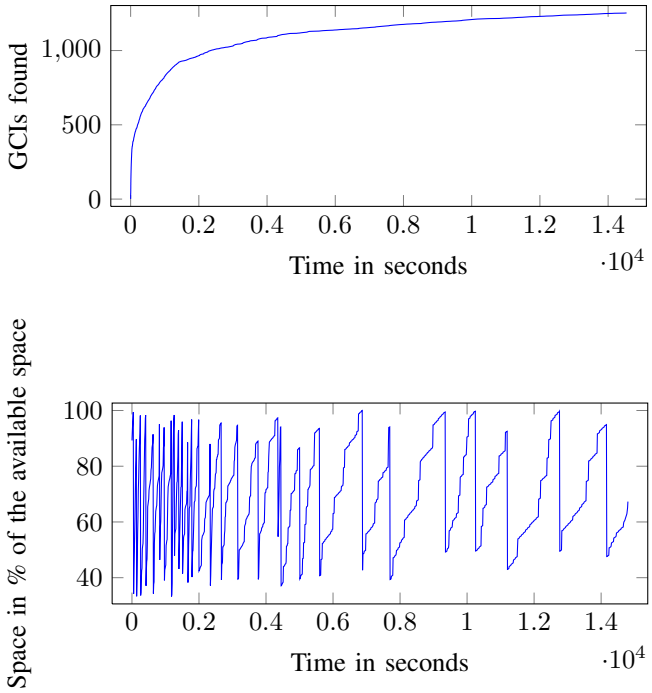

Fig. 2: Time and space behavior when exploring the DBpedia-model.

relies on the computation of pseudo intents in lexicographic order (using Next-Closure), this bottleneck can most likely not be avoided [14]. Unfortunately, our experiments indicate that the worst case might be very close to the average case in our application.

For the space behavior of our implementation concrete interpretations are much harder to make. This is because of the Java Virtual Machine and its memory model on which the implementation runs on, which is based on automatic garbage collection. However, from the description of the algorithm we can expect a very moderate memory consumption of the overall run of the exploration, which should mostly be due to storing the generated GCIs. All other operations involved might need some extra space but only for a short time. This expectation can be seen from the graphical representation because of the appearance of "stairs" in the diagram, showing the periods where the next GCI is searched for, alternating with the periods of steep increase, where other computations are performed. The steep declines of the curve correspond to calls of the garbage collector and they show that in almost every case that 50% of the main memory could have been reclaimed, further supporting our hypothesis that only a moderate number of memory is used.

Finally, it is interesting to see that the calls to the garbage collector get less and less the longer the exploration runs. This corresponds to the fact that it takes more and more time (in average) to compute the next GCI the longer the exploration runs.

V. Future Work

In the future two main issues should be addressed. First, we would like to investigate approaches to improve the quality of the output, in particular in the case of incomplete or noisy data. Second, we want to improve on memory and space requirements.

A. Improving the Quality of the Output

One should not forget that the algorithms that are used here are mathematically proven to yield a sound and complete base of minimal cardinality for the dataset provided. That means, provided that the data is complete and error-free there is no room for improvement in terms of quality. In practice, we are unlikely to ever encounter perfect data. Hence the question is, how can the effects of poor data be mitigated.

In Section IV-C1 we have observed the two different types of influence of incomplete data and faulty data. In association rule mining these two problems are usually dealt with by introducing the measures of *support* and *confidence* [2]. A naive translation of the support and confidence to the terminology of $\mathcal{EL}$-GCIs could look like this:

$$\operatorname{supp}(A \sqsubseteq B) = \frac{|(C \sqcap D)^I|}{\Delta_I} \quad \operatorname{conf}(C \sqsubseteq D) = \frac{|(C \sqcap D)^{\mathcal{I}}|}{|C^I|} \tag{10}$$

The idea is that by searching only for GCIs with a certain minimal support one can mitigate the effects of incomplete data. At the same time imposing a minimum on the confidence can limit the effects of noise in the data.

Several obstacles need to be overcome to make this approach work. First, we have argued that the large number of possible GCIs requires a compact representation, e.g. a base. Now, if $A \sqsubseteq B$ and $B \sqsubseteq C$ both have a confidence greater than 0.9, the GCI $A \sqsubseteq C$, even though it follows from the first two, may have a confidence as low as 0.81. Hence, imposing a lower limit on the confidence results in a set of GCIs that is no longer closed with respect to semantic deductions. In such a situation computing a base is meaningless. An obvious way to mend this is to consider both GCIs with sufficient confidence and support, as well as all GCIs that follow semantically from them to be trustworthy. This set of trustworthy GCIs has the necessary closure property, but has not been researched theoretically.

An alternative approach is to look for other compact representations of the GCIs with sufficient support. One might think of an analogous notion to frequent closed itemsets (e.g. frequent "closed" $\mathcal{EL}$-concept descriptions) and Iceberg lattices [17].

We also need to address whether the naive translation of the classical notions of support and confidence really does what one intuitively expects it to. As a toy example assume that a dataset about art is very incomplete, containing only one Museum, the Louvre, but thousands of pieces of art that belong to the Louvre (cf. Figure 3). Clearly, the GCI

$$\mathsf{Museum} \sqsubseteq \exists \mathsf{hasLocation}.\mathsf{FrenchCity},$$

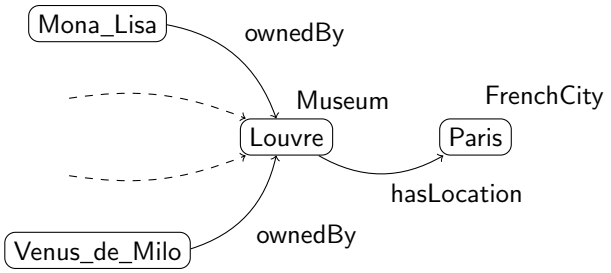

Fig. 3: Classical Support is Counterintuitive

which states that all museums are in a French city, is no more trustworthy than the GCI

$$\exists \mathsf{ownedBy}.\mathsf{Museum} \sqsubseteq \exists \mathsf{ownedBy}.(\exists \mathsf{hasLocation}.\mathsf{FrenchCity}),$$

which states that something that is owned by a museum is owned by a museum in a French city. While the first GCI has very small support (there is only one museum in the dataset) the latter has very large support (in the dataset there are thousands of pieces of art belonging to a museum). Whether other notions of support are less counterintuitive needs to be investigated.

B. Speeding Up the Mining Process

In its current state, the runtime and the memory requirements of our implementation leave plenty of room for improvement. While it is unlikely that the bottleneck (with respect to runtime) that is presented by Next-Closure can be avoided completely, there are some improvements that can be made. There is hope that the introduction of a concept constructor $\bot$ describing the empty concept will significantly improve memory requirements by making the All description redundant. All is a very large concept description and since it occurs quite frequently a large amount of memory is quickly filled with various instances of All.

Another rather technical improvement would be to avoid computing large intermediate results. This is mostly due to computing large description graphs which then turn out to be highly redundant. It would be more desirable to directly compute a smaller part of this description graph which is sufficient for our purposes. Improvements in this direction would not only decrease the memory requirements for the implementation but also the overall runtime, because extra computation for reducing description graphs could be avoided.

As a rather drastic measure one could impose an upper bound on the role depth of the $\mathcal{EL}$-descriptions. This would both remove the need for cyclic concept descriptions and eliminate concept descriptions that consume large amounts of memory. Our experiments provide anecdotal evidence that GCIs with larger role depths are usually among the least interesting.

C. Quality Assessment

The computed GCIs are sound and complete for the given data set. However, that neither means that they are correct

in the domain nor that they are complete therein. Therefore the results computed should merely be regarded as a starting point for constructing of a knowledge base. Whether the quality of the GCIs that are produced is sufficient in practical applications should be examined in a case study in cooperation with domain experts.

VI. Conclusion

We have prototypically implemented an algorithm that is able to mine a small set of $\mathcal{EL}$-GCIs from a dataset. We have tested this algorithm on two real world datasets. The results, at least for the relatively error-free dataset of the DrugBank, appear promising. However, for a final verdict, a quality assessment with domain experts will be needed. We have also observed and discussed the implications of incomplete and faulty data, as they appeared in the DBpedia dataset. To the purpose of improving the quality of the GCIs when mining from imperfect data, we have suggested to look into measures similar to support and confidence, which are known from association rule mining.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, May 1993, pp. 207–216.

[3] F. Distel, "Learning description logic knowledge bases from data using methods from formal concept analysis," Ph.D. dissertation, TU Dresden, Dresden, Germany, June 2011.

[4] F. Baader and F. Distel, "Exploring finite models in the Description Logic $\mathcal{EL}_{\mathrm{gfp}}$," in *Proc. of the 7th Int. Conf. on Formal Concept Analysis (ICFCA 2009)*, S. Ferré and S. Rudolph, Eds. Springer, 2009.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *In 6th Intl Semantic Web Conference, Busan, Korea*. Springer, 2007, pp. 11–15.

[6] D2R server publishing the drugbank database. FU Berlin. [Online]. Available: http://www4.wiwiss.fu-berlin.de/drugbank/

[7] the data hub. Comprehensive Knowledge Archive Network. [Online]. Available: http://ckan.net/

[8] K. Spackman, K. Campbell, and R. Cote, "SNOMED RT: A reference terminology for health care," pp. 640–644, 1997, fall Symposium Supplement.

[9] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene Ontology: Tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[10] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, "Owl 2 web ontology language: Profiles," W3C Recommendation, October 2009.

[11] F. Baader, B. Ganter, U. Sattler, and B. Sertkaya, "Completing Description Logic knowledge bases using Formal Concept Analysis," in *IJCAI-07*, 2007.

[12] S. Rudolph, "Relational exploration – combining Description Logics and formal concept analysis for knowledge specification," Ph.D. dissertation, Technische Universität Dresden, 2006.

[13] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. New York: Springer, 1997.

[14] F. Distel, "Hardness of enumerating pseudo-intents in the lectic order," in *Proc. of the 8th Int. Conf. on Formal Concept Analysis (ICFCA 2010)*, ser. Lecture Notes in Artificial Intelligence, B. Sertkaya and L. Kwuida, Eds., vol. 5986. Springer, 2010, pp. 124–137.

[15] D. Borchmann, "Implementing the exploration algorithm for ELgfp," TU Dresden, Tech. Rep., 2011.

[16] D2R server publishing the diseasome dataset. FU Berlin. [Online]. Available: http://www4.wiwiss.fu-berlin.de/diseasome/

[17] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal, "Computing iceberg concept lattices with TITANIC," *Data Knowl. Eng.*, vol. 42, no. 2, pp. 189–222, 2002.