An Approach to Exploring Description Logic Knowledge Bases

Felix Distel

Theoretical Computer Science, TU Dresden, Germany
felix@tcs.inf.tu-dresden.de

Abstract. This paper is the successor to two previous papers published at the ICFCA conference. In the first paper we have shown that in the Description Logics $\mathcal{EL}$ and $\mathcal{EL}_{\text{gfp}}$, the set of general concept inclusions holding in a finite model always has a finite basis. An exploration formalism that can be used to obtain this basis was presented in the second paper. In this paper we show how this formalism can be modified such that counterexamples to GCIs can be provided in the form of ABox-individuals. In a second part of the paper we examine which description logics can be used for this ABox.

1 Introduction

Description Logics (DLs) are a formalism for representing knowledge that has gained international recognition during the last decade [3]. They play a significant role in the Semantic Web Community, in particular because of the OWL language which is essentially a variant of an expressive DL [10].

A DL knowledge base usually consists of two parts. The first part, the TBox is used to describe the terminology of the knowledge base. It contains general concept inclusion (GCIs), i. e. statements of the form $C \sqsubseteq D$. Here C and D are concept descriptions written using a set of so-called concept constructors, concept names and role names. Different DL languages use different concept constructors. However, all DL languages provide a formal, well-defined model based semantics for the concept descriptions. A model $i = (\Delta_i, \cdot^i)$ consists of a set Δ_i and a function $\cdot^i$ that maps concept descriptions C to subsets $C^i \subseteq \Delta_i$. The second part of the knowledge base is the ABox. It contains knowledge about individuals. One can for example assert that an individual `Henry` belongs to the

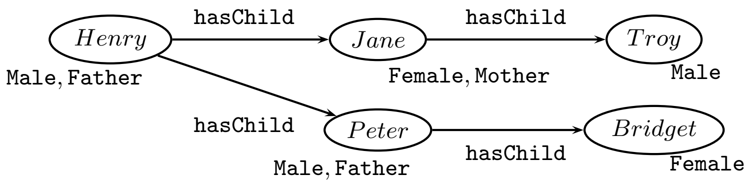

Fig. 1. A Model of a Family of Three Generations

concept `Father` or that there is a `hasChild` role leading from `Henry` to `Jane`. An important aspect are the open world semantics of ABoxes. If it is not stated that `Henry` is a `Father` then it is not assumed that `Henry` is not a `Father`.

Writing knowledge bases can be a difficult process, in particular because experts in the domain of the knowledge base are usually not experts in DL. In order to help them to find the right GCIs to add to their TBox, one approach is to use a formalism that is inspired by attribute exploration from Formal Concept Analysis (FCA) [9]. In the formalism that has been presented in a previous ICFCA Paper [6] it is assumed that the domain of the knowledge base can be represented as a DL model, and that this model is completely known to a human expert. In this formalism the expert does not have to come up with GCIs herself. Instead the system suggests GCIs that she can either add to the TBox, or reject by providing a counter-example. This approach is often referred to as *knowledge base completion*.

Let us assume that the domain was represented by the model of a family of three generations from Figure 1. The system might come up with a GCI like $\texttt{Father} \sqsubseteq \texttt{Male} \sqcap \exists\texttt{hasChild}.\top$, i. e. "Every father is male and has a child." The expert would obviously accept this GCI and add it to the knowledge base. If, however, the system comes up with the GCI $\texttt{Father} \sqsubseteq \texttt{Mother}$, i. e. "Every father is a mother", then the expert would reject it and add e. g. `Henry` as a counter-example.

The GCIs from the example are written in the lightweight description logic $\mathcal{EL}$. $\mathcal{EL}$ is less expressive than most other standard DLs but has the advantage that standard reasoning tasks are tractable [8]. This is one of the reasons why tractable extensions of $\mathcal{EL}$ are used for large scale biomedical ontologies such as SNOMED [12] and the Gene Ontology [13].

Our algorithm from the previous ICFCA paper also uses a tractable extension of $\mathcal{EL}$, $\mathcal{EL}_{\text{gfp}}$, which allows the algorithm to generate concept descriptions that are cyclic. The major weakness of our previous algorithm is the way in which counter-examples are provided. It uses connected submodels which use a closed-world semantics. The submodel is extended every time the expert provides a counter-example. Let us assume the expert wants to state that `Henry` is a counter-example to the GCI $\texttt{Father} \sqsubseteq \texttt{Mother}$. Assume that the expert only adds `Henry`, but not `Jane` or `Peter`, to the submodel and says that `Henry` is a `Father` but not a `Mother`. Because of the closed world semantics the algorithm would assume that `Henry` does not have children which would make `Henry` a counter-example to the GCI $\texttt{Father} \sqsubseteq \texttt{hasChild}.\top$. This is unwanted because $\texttt{Father} \sqsubseteq \texttt{hasChild}.\top$ does hold in the domain. The only way to avoid this effect is to add not only `Henry`, but also all of his direct or indirect role successors, in this case his children and grandchildren.

So the expert would need to add a lot more information than is actually needed to make `Henry` a counter-example without creating unwanted artefacts. This is inconvenient and can only be overcome by allowing open-world-semantics. In the DL-world the natural datastructure to keep track of individuals which provides an open-world semantics is an ABox. This paper will present an approach how to extend the algorithm from the previous paper to work with ABoxes as the underlying datastructure. We will introduce minimal possible consequences as a central notion. Since this is ongoing work some important questions remain open, e. g. if and how minimal possible consequences can be computed effectively.

Due to space restrictions we cannot introduce Formal Concept Analysis. We assume that the reader is familiar with the basic notions from this field.

Related Work: There are two other works important works that try to combine FCA and DL. The work by Baader et al. provides a knowledge base completion formalism that also uses ABoxes as the underlying datastructure [7]. However, their algorithm does not perform knowledge base completion with respect to arbitrary GCIs written in a language like $\mathcal{EL}$. Instead they only allow conjunctions of previously defined concepts. The second approach by Rudolph can be used to compute a basis for the GCIs of a given DL model. The main difference compared to our approach lies in the way the GCIs are computed. While we construct a context on the fly, adding only a few interesting attributes at a time, Rudolph's approach successively increases role depth and adds all attributes up to a certain depth [11].

2 Preliminaries

The Description Logic $\mathcal{EL}$ Due to space restrictions we can only give a brief introduction to the DLs $\mathcal{EL}$ and $\mathcal{EL}_{\text{gfp}}$. $\mathcal{EL}$ concept descriptions are generated from a finite set $\mathcal{N}_C$ of concept names and a finite set $\mathcal{N}_r$ of role names as follows.

- concept names and the top concept $\top$ are $\mathcal{EL}$-concept descriptions;
- if C, D are $\mathcal{EL}$-concept descriptions and r is a role name, then $C \sqcap D$ and $\exists r.C$ are $\mathcal{EL}$-concept descriptions.

The tuple $\Sigma = (\mathcal{N}_C, \mathcal{N}_r)$ is called the *signature* of the concept description.

A $\mathcal{EL}$ model $i = (\Delta_i, \cdot^i)$ consists of a finite set Δ_i, the so-called domain of the model, and an interpretation function $\cdot^i$ mapping role names r to relations $r^i \subseteq \Delta_i \times \Delta_i$ and concept descriptions C to their extensions such that

$$\top^i = \Delta_i, \quad (C_1 \sqcap C_2)^i = C_1^i \cap C_2^i, \quad \text{and} \\ (\exists r.D)^i = \{d \in \Delta_i \mid \exists e \in D^i \text{ such that } (d,e) \in r^i\}.$$

Note that it suffices to define the interpretation function for role names and concept names. The interpretations of more complex concept descriptions can then be derived, recursively. Subsumption and equivalence between $\mathcal{EL}$-concept descriptions is defined in the usual way, i.e., C is subsumed by D (written $C \sqsubseteq D$) iff $C^i \subseteq D^i$ for all models i, and C is equivalent to D (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$.

TBoxes and ABoxes A GCI is a statement of the form $C \sqsubseteq D$, where C and D are concept descriptions. We say that a GCI $C \sqsubseteq D$ *holds in a model* i if $C^i \subseteq D^i$ holds. Note that this is not the same as subsumption. An equivalence statement is a statement of the form $A \equiv D$, where A is a concept name and D a concept description. $A \equiv D$ is said to *hold in* i if $A^i = D^i$.

TBoxes are sets of equivalence statements and GCIs. They fall into three categories.

– *Acyclic TBoxes* contain only equivalence statements where the left-hand side is not used in the concept description on the right-hand side implicitly or explicitly.
– *Cyclic TBoxes* contain only equivalence statements
– *General TBoxes* contain arbitrary GCIs.

A model i is said to be a model of a TBox $\mathcal{T}$ if all statements from $\mathcal{T}$ hold in i. In the case of cyclic TBoxes there exists also the notion of *greatest-fixpoint-models.* Informally, a model i is a greatest-fixpoint model of $\mathcal{T}$ if the interpretations of all concept names in i are maximal among all other models of $\mathcal{T}$ with the same domain. A more formal definition can be found in [2].

An *ABox* $\mathcal{A}$ is a set of concept assertions and role assertions, where a *role assertion* is of the form $r(a, b)$ and a *concept assertion* is of the form $A(a)$, with r a role name, A a concept name, and a and b so-called individual names. A *model* $i = (\Delta_i, \cdot^i)$ *of an ABox* $\mathcal{A}$ is a model where $\cdot^i$ is extended to map individual names a to individuals $a^i \in \Delta_i$ such that $a \in A^i$ for all concept assertions $A(a) \in \mathcal{A}$ and $(a, b) \in r^i$ for all role assertions $r(a, b) \in \mathcal{A}$.

The Description Logic $\mathcal{EL}_{\mathbf{gfp}}$ $\mathcal{EL}_{\text{gfp}}$ is the extension of $\mathcal{EL}$ by cyclic concept definitions interpreted with greatest fixpoint (gfp) semantics. In $\mathcal{EL}_{\text{gfp}}$, we assume that the set of concept names is partitioned into the set $\mathcal{N}_{\text{prim}}$ of primitive concepts and the set $\mathcal{N}_{\text{def}}$ of defined concepts. We only allow concept definitions of the form

$$B_0 \equiv P_1 \sqcap \ldots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \ldots \sqcap \exists r_n.B_n \tag{1}$$

where $B_0, B_1, \ldots, B_n \in \mathcal{N}_{\text{def}}$, $P_1, \ldots, P_m \in \mathcal{N}_{\text{prim}}$, and $r_1, \ldots, r_n \in \mathcal{N}_r$. The empty conjunction (i.e., $m = 0 = n$) stands for $\top$.

Definition 1 **($\mathcal{EL}_{\text{gfp}}$-concept description).** *A* $\mathcal{EL}_{\text{gfp}}$-concept description *is a tuple* $(A, \mathcal{T})$ *where* $\mathcal{T}$ *is a TBox and* A *is a defined concept occurring on the left-hand side of a definition in* $\mathcal{T}$*.*

Let $i = (\Delta_i, \cdot^i)$ be a model. The *extension* $(A, \mathcal{T})^i$ *of* $(A, \mathcal{T})$ in i is the set assigned to A by the gfp-model of $\mathcal{T}$ based on i. Subsumption and equivalence between $\mathcal{EL}_{\text{gfp}}$-concept descriptions is defined as in the case of $\mathcal{EL}$-concept descriptions. It is easy to see that acyclic $\mathcal{EL}_{\text{gfp}}$-concept descriptions (i.e., ones where the TBox component is acyclic) correspond exactly to $\mathcal{EL}$-concept descriptions.

It is difficult to obtain a good intuition about greatest-fixpoint semantics. Fortunately, there is an alternative characterization. Given a model i and an individual $x \in \Delta_i$ we can define the set of concept names assigned to x as $\text{names}_i(x) = \{A \in \mathcal{N}_{\text{prim}} \mid x \in A^i\}$. We denote the set of all r-successors of x in i by $xr^i = \{y \in \Delta_i \mid (x, y) \in r^i\}$.

For TBoxes that contain only concept definitions of the form (1) we introduce notations similar to those for models. For a defined concept B we denote by $\text{names}_{\mathcal{T}}(B)$ the set of all primitive concept names $P_1, \ldots, P_k$ that occur in the definition of B in $\mathcal{T}$. For a defined concept B_1 and a role name r we denote by $B_1 r_{\mathcal{T}}$ the set of all defined concept names B_2 for which the term $\exists r.B_2$ occurs in the definition of B_1 in $\mathcal{T}$. A simulation from a normalized TBox $\mathcal{T}$ to a model i is a relation $\zeta \subseteq \mathcal{N}_{\text{def}} \times \Delta_i$ where

(S1) $\text{names}_{\mathcal{T}}(B) \subseteq \text{names}_i(x)$ for all pairs $(B, x) \in \zeta$. , and
(S2) for all role names $r \in \mathcal{N}_r$, all pairs $(B, x) \in \zeta$ and all $E \in Br_{\mathcal{T}}$ there is some $y \in xr^i$ such that $(E, y) \in \zeta$ holds.

The following theorem enables us to check instance without using greatest fixpoints explicitly [1].

Lemma 1. *Let $C = (A_C, \mathcal{T}_C)$ be an $\mathcal{EL}_{\text{gfp}}$-concept description. Let $i = (\Delta_i, \cdot^i)$ be a model and $x \in \Delta_i$ an individual. Then it holds that $x \in C^i$ iff there is a simulation ζ from $\mathcal{T}_C$ to i that contains (A_C, x).*

Given a set of GCIs $\mathcal{B}$, we say that the GCI $C \sqsubseteq D$ follows from $\mathcal{B}$ if $C \sqsubseteq D$ holds in all models in which all GCIs from $\mathcal{B}$ hold. We say that $\mathcal{B}$ is a *basis* for the $\mathcal{EL}_{\text{gfp}}$-GCIs holding in i if $\mathcal{B}$ is

- *sound* for i, i. e. it contains only GCI that hold in i, and
- *complete* for i, i. e. any $\mathcal{EL}_{\text{gfp}}$-GCI holding in i follows from $\mathcal{B}$.

3 Results from Previous Work

Exploration as a method for knowledge base completion relies on the existence of an expert with complete knowledge about the domain of the knowledge base. For practical purposes we assume that the domain of the knowledge base ("the real world") can be represented as a model i of the final (complete) knowledge base. In this and the previous work i is called the *background model.* The goal of an exploration is to find a basis for the set of GCIs holding in i.

In doing this we face two major challenges: First, for most DLs it is not trivial to find a basis, even when the background model is known. Second, since the complete background model is unknown to the algorithms, the algorithm must gradually gain information about the background model by querying the expert. The first challenge has been adressed in [5] while a solution for the second problem is proposed in [6]. The purpose of this section is to recapitulate important notions from these two publications.

3.1 Model-Based Most Specific Concepts

Suppose we want to compute a basis for the GCIs holding in the model i from Figure 1. Suppose furthermore that we have decided (for example by using the algorithm described in [6]) that the $\mathcal{EL}_{\mathrm{gfp}}$-concept description `Father` is an interesting premise for a GCI. We might add any of the GCIs `Father` $\sqsubseteq$ `Male`, or `Father` $\sqsubseteq$ `Male` $\sqcap \exists$`hasChild`.$\top$ to the basis. However, if we decide to add the first one and later find out that we need to add also the second to ensure completeness we obtain redundance (because the first GCI follows from the latter). In an exploration setting this would mean that we would ask two questions where one would be enough. To avoid redundant questions, the idea is to be as specific as possible when choosing the right-hand side of a GCI. For the description logic $\mathcal{EL}_{\mathrm{gfp}}$ model-based most specific concepts are what we need to find these conclusions.

Definition 2 (Model-Based Most Specific Concept). *Let $i = (\Delta_i, \cdot^i)$ be a finite model and $X \subseteq \Delta_i$ a set. The $\mathcal{EL}_{\mathrm{gfp}}$-concept description C is the* most specific $\mathcal{EL}_{\mathrm{gfp}}$-concept of X in i *if it is the least $\mathcal{EL}_{\mathrm{gfp}}$-concept description such that $X \subseteq C^i$. By* least $\mathcal{EL}_{\mathrm{gfp}}$-concept description *we mean that every other $\mathcal{EL}_{\mathrm{gfp}}$-concept description $\bar{C}$ satisfying $X \subseteq \bar{C}^i$ also satisfies $C \sqsubseteq \bar{C}$.*

It is justified to speak of *the* model-based most specific concept (mmsc) because the model-based most specific concept is unique up to equivalence. We use the notation X^i to denote the mmsc of X. Mmsc for the description logic $\mathcal{EL}_{\mathrm{gfp}}$ exist for all models $i = (\Delta_i, \cdot^i)$ and all sets $X \subseteq \Delta_i$ and can be computed effectively [5].

Lemma 2. *Let i be a model, $X, Y \in \Delta_i$ sets of objects and let C, D be $\mathcal{EL}_{\mathrm{gfp}}$-concept descriptions. Then the following statements hold*

1. $X \subseteq Y \Rightarrow X^i \sqsubseteq Y^i$
2. $C \sqsubseteq D \Rightarrow C^i \subseteq D^i$
3. $X \subseteq X^{ii}$
4. $C^{ii} \sqsubseteq C$
5. $X^i \equiv X^{iii}$
6. $C^i = C^{iii}$
7. $X \subseteq C^i \Leftrightarrow X^i \sqsubseteq C$.

This lemma from [5] shows that GCIs of the form $C \sqsubseteq C^{ii}$ play a special rôle.

Lemma 3. *Let C, D be $\mathcal{EL}_{\mathrm{gfp}}$-concept descriptions and i a finite $\mathcal{EL}_{\mathrm{gfp}}$-model. Then $C \sqsubseteq C^{ii}$ holds in i. If $C \sqsubseteq D$ holds in i, then $C \sqsubseteq D$ follows from $\{C \sqsubseteq C^{ii}\}$.*

We have seen that mmsc can help to reduce redundancy. They are therefore useful when it comes to constructing finite sets of axioms for a given model.

3.2 An Algorithm for Axiomatizing a Given Model

In [6] an algorithm has been presented that can be used to axiomatize a given (known) model i (Algorithm 1). Given a finite model i as input Algorithm 1 will

Algorithm 1 Computing a basis for an a priori given model i

```
 1: Input: finite model $i = (\Delta_i, \cdot^i)$
 2: $M_0 := \mathcal{N}_C$, $\mathcal{S}_0 := \emptyset$
 3: $\Pi_0 := \emptyset$, $P_0 := \emptyset$, $k := 0$
 4: while $P_k \neq \mathsf{null}$ do
 5:     $\Pi_{k+1} := \Pi_k \cup \{P_k\}$
 6:     $M_{k+1} := M_k \cup \{\exists r.(\bigsqcap P_k)^{ii} \mid r \in \mathcal{N}_r\}$
 7:     $\mathcal{S}_{k+1} := \{\{C\} \to \{D\} \mid C, D \in M_{k+1}, C \sqsubseteq D\}$
 8:     $k := k + 1$
 9:     if $M_k = M_{k-1} = P_k$ then
10:         $P_k := \mathsf{null}$
11:     else
12:         $P_k :=$ lectically next set of attributes that respects all implications in
                $\{P_j \to P_j^{\prime\prime_k} \mid 1 \leq j < k\}$ and $\mathcal{S}_k$
13:     end if
14: end while
```

always terminate. Upon termination it will have produced a set Π_n of so-called premises P_k such that

$$\mathcal{B}_n := \{\bigsqcap P_k \sqsubseteq (\bigsqcap P_k)^{ii} \mid P_k \in \Pi_n\}$$

is a basis for the GCIs holding in i.

The algorithm uses the notation $\bigsqcap U$, where U is a set of concept descriptions, to denote the concept $\bigsqcap U := \bigsqcap_{D \in U} D$.

It uses some elements of FCA, in particular the next-closure algorithm. The connection between FCA and DL is made by so-called induced contexts. What we call induced contexts in this work are formal contexts whose attributes are concept descriptions and whose set of objects is the domain Δ_i of a finite model i. More formally, let i be a finite $\mathcal{EL}_{\text{gfp}}$-model and M a finite set of $\mathcal{EL}_{\text{gfp}}$-concept descriptions. The *context induced by M and i* is the formal context $\mathbb{K} = (G, M, I)$, where $G = \Delta_i$ and $I = \{(x, C) \mid C \in M \text{ and } x \in C^i\}$.

There are infinitely many possible concept descriptions and thus infinitely many possible attributes for an induced context. The most important idea in the construction of Algorithm 1 was that the set of attributes was not fixed in the beginning. Instead a new set of attributes M_k is generated during each iteration. The notation $\cdot^{\prime\prime_k}$ denotes the $\cdot''$-operator from FCA computed in the context $\mathbb{K}_k$, where $\mathbb{K}_k$ denotes the context induced by M_k and i.

3.3 Exploration Using Submodels

Of the two main challenges that we have identified, the second one was constructing a set of axioms in a situation where the background model is not known to the algorithm. The only way to gain information about the model is to ask the expert. In [6] an algorithm has been presented that uses the familiar exploration principle. It generates a GCI and asks the expert whether this GCI holds in the background model. If so, the GCI is added to the set of axioms. Otherwise the expert is asked to provide a counterexample. Now the question is in what form these counterexamples should be provided. In [6] the counterexamples are provided in the form of connected submodels of the background model.

Thereby a submodel j of a model i is a model such that $\Delta_j \subseteq \Delta_i$ and $C^j = C^i \cap \Delta_j$ for all concept names C and $r^j = r^i \cup (\Delta_j \times \Delta_j)$ for all role names r. j is called a *connected submodel* if and only if for every $x \in \Delta_i$ and all $r \in \mathcal{N}_r$ if $x \in \Delta_j$ then all r-successors of x are also in Δ_j. Whenever a GCI is refuted the expert is asked to provide a new model i_j that we call the *working model.* It is required to extend the previous working model i_{j-1}, to be a connected submodel of i and to contain a counterexample. Similar to Algorithm 1 it has been shown that this algorithm always terminates and produces a basis for the set of implications holding in i.

4 Replacing Models by ABoxes

4.1 Possible Consequences

We consider a setting where (instead of a connected submodel of the background model i) the expert provides a knowledge base consisting of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. For now, the background model i should be a model of the ABox $\mathcal{A}$ that contains the counterexamples and the TBox $\mathcal{T}$. Given $\mathcal{A}$ and $\mathcal{T}$ what can be said about the GCIs that hold in i? First, there are the GCIs that hold in *every* model of $\mathcal{A}$ and $\mathcal{T}$. These are the GCIs which are already known to hold in i. Therefore they are not interesting for a completion formalism.

On the other hand, there are the GCIs that hold in at least one model of $\mathcal{A}$ and $\mathcal{T}$. Since the background model i is unknown, it is possible that i is one of these models in which the GCI holds. So these GCIs are the ones we are interested in. Provided an $\mathcal{EL}_{\text{gfp}}$-concept description C we define the set of concept descriptions D that are *possible consequences* of C to be

$$\mathfrak{pc}_{\mathcal{A},\mathcal{T}}(C) = \{D \mid \exists j \text{ model of } \mathcal{A} \text{ and } \mathcal{T} : C^j \subseteq D^j\}.$$

Notice, that we do not make any requirements with respect to the language of the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$, except that they have a model-theoretic semantics with models as defined in Section 2. It may be different from $\mathcal{EL}_{\text{gfp}}$. The certain and possible consequences, however, are expressed in $\mathcal{EL}_{\text{gfp}}$.

Now, suppose we want to present to the expert a GCI $C \sqsubseteq D$ whose left-hand side is C. It does not make sense to ask this question, unless D is a possible consequence of C. Otherwise the answer would certainly be "No". So we have to choose D among the possible consequences of C.

Once the expert accepts a GCI, the algorithm should not have to generate another GCI with the same premise. This is why we introduce the notion of minimal possible consequences. D is said to be a minimal possible consequence of C if $D \in \mathfrak{pc}_{\mathcal{A},\mathcal{T}}(C)$ and D is minimal in $\mathfrak{pc}_{\mathcal{A},\mathcal{T}}(C)$ with respect to $\sqsubseteq$. The set of all minimal possible consequences of C is denoted by $\mathfrak{mpc}_{\mathcal{A},\mathcal{T}}(C)$. Unlike mmsc minimal possible consequences need not be unique up to equivalence. We will mostly be interested in GCIs over a fixed signature Σ. We introduce the notation $\mathfrak{pc}^{\Sigma}_{\mathcal{A},\mathcal{T}}(C)$ for the set of all possible consequences that are expressed using only the signature Σ. Analogously, we define $\mathfrak{mpc}^{\Sigma}_{\mathcal{A},\mathcal{T}}(C)$.

Those who are familiar with [7] will find that the $\mathcal{K}(\cdot)$-operator computes minimal possible consequences for the special case of a logic that allows only for conjunction.

4.2 Adapting the Exploration Algorithm

It is not yet known if (or rather for which logics) minimal possible consequences exist. This is work in progress and will not be considered here. For now, we assume that the knowledge bases considered here are written in a logic for which the existence of minimal possible consequences is guaranteed. We also assume that there exists an oracle to compute a minimal possible consequence for a given $\mathcal{EL}_{\mathrm{gfp}}$-concept description C.

We show that under these assumptions Algorithm 1 requires only subtle modifications in order to function with ABoxes as underlying datastructure. The modified algorithm is presented as Algorithm 2. We assume that there is a background model i which is known to the expert. The input consists of a TBox $\mathcal{T}_0$ and an ABox $\mathcal{A}_0$ (instead of a model). We require that i is a model of the $\mathcal{T}_0$ and $\mathcal{A}_0$. The signature of the initial knowledge base is denoted by Σ_0.

The modification with respect to Algorithm 1 primarily consists in the addition of a second while-loop. Informally, the purpose of this inner while-loop is to find the proper conclusion D_k to a given premise $\bigsqcap P_k$. Since i is not explicitly given it is not possible to directly compute $(\bigsqcap P_k)^{ii}$ like in Algorithm 1.

Before we start to prove completeness, let us first clarify a few details about Algorithm 2. First of all, note that while the newly acquired GCIs (i. e. the P_k found in the algorithm) are formulated in $\mathcal{EL}_{\mathrm{gfp}}$ we do not specify the logic of the underlying ABox and TBox. Using two different languages may seem unnatural at first, but is, unfortunately, necessary. This will become clear in Section 5.

In Line 19 $\mathrm{pr}_M(C)$ denotes the *projection* of a concept description C to a set of concept descriptions M, i. e. the set $\mathrm{pr}_M(C) = \{D \in M \mid C \sqsubseteq D\}$. The following lemma about projections in induced contexts has been proved in [4].

Lemma 4. *Let $U \subseteq M$ be any set of attributes in a context $\mathbb{K}$ induced by i and M. Then $U'' = \mathrm{pr}_M\left((\bigsqcap U)^{ii}\right)$.*

A last remark concerns the changing signatures. In Line 8 the expert is asked to provide a new TBox $\mathcal{T}_j$ and ABox $\mathcal{A}_j$. We allow that new concept names that are not present in Σ_0 are used in $\mathcal{T}_j$ and $\mathcal{A}_j$. The motive behind this is that in certain logics new concept names are necessary to express that an individual is a counterexample to a GCI (cf. Section 5.2). Allowing new concept names yields one problem: It is not clear how to interpret the new concept names in the background model. In other words i is not a model of $\mathcal{T}_j$ and $\mathcal{A}_j$. That is why we introduce the notion of a *representation* of a model. $\mathcal{T}_j$ and $\mathcal{A}_j$ are called a representation of i if there is a model ι of $\mathcal{T}_j$ and $\mathcal{A}_j$ such that

Algorithm 2 The ABox Exploration Algorithm

```
 1: Input: ABox $\mathcal{A}_0$, TBox $\mathcal{T}_0$ with signature $\Sigma_0$
 2: $M_0 := \mathcal{N}_{\text{prim}}$, $\mathcal{S}_0 := \emptyset$
 3: $\Pi_0 := \emptyset$, $P_0 := \emptyset$, $k := 0$, $j := 0$
 4: while $P_k \neq \mathsf{null}$ do
 5:     Obtain $D \in \mathtt{mpc}^{\Sigma_0}_{\mathcal{T}_j,\mathcal{A}_j}(\bigsqcap P_k)$ from oracle
 6:     while expert refutes $\bigsqcap P_k \sqsubseteq D$ do
 7:         $j := j + 1$
 8:         Ask the expert for a new knowledge base $(\mathcal{T}_j, \mathcal{A}_j)$ that extends $(\mathcal{T}_{j-1}, \mathcal{A}_{j-1})$,
            and is a representation of $i$.
 9:         Obtain $D \in \mathtt{mpc}^{\Sigma_0}_{\mathcal{T}_j,\mathcal{A}_j}(\bigsqcap P_k)$ from oracle
10:     end while
11:     $D_k = D$
12:     $\Pi_{k+1} := \Pi_k \cup \{P_k\}$
13:     $M_{k+1} := M_k \cup \{\exists r.D_k \mid r \in \mathcal{N}_r\}$
14:     $\mathcal{S}_{k+1} := \{\{C\} \to \{E\} \mid C, E \in M_{k+1}, C \sqsubseteq E\}$
15:     $k := k + 1$
16:     if $M_k = M_{k-1} = P_k$ then
17:         $P_k := \mathsf{null}$
18:     else
19:         $P_k :=$ lectically next set of attributes that respects all implications in
                 $\{P_l \to \mathrm{pr}_{M_k}(D_l) \mid 1 \leq l < k\}$ and $\mathcal{S}_k$
20:     end if
21: end while
```

- $\Delta_i = \Delta_\iota$, and
- for all $\mathcal{EL}_{\text{gfp}}$-concept descriptions over the smaller signature Σ_0 it holds that $C^i = C^\iota$.

Once the expert has accepted a GCI, the algorithm should not need to consider the same premise P_k again. Lemma 5 shows why this is indeed the case.

Lemma 5. *Whenever Algorithm 2 leaves the inner while-loop (Lines 6 to 10) it holds that* $D \equiv (\bigsqcap P_k)^{ii}$.

Proof. The algorithm will only leave the inner while-loop when the expert states that $\bigsqcap P_k \sqsubseteq D$ holds in i. This means that $(\bigsqcap P_k)^i \subseteq D^i$ is true. Lemma 2 implies that $(\bigsqcap P_k)^{ii} \sqsubseteq D$. Because $\mathcal{A}_j$ and $\mathcal{T}_j$ are a representation, there must be a model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ such that for all $\mathcal{EL}_{\text{gfp}}$-concept descriptions C over the smaller signature Σ_0 it holds that $C^i = C^\iota$. Since $\bigsqcap P_k$ and $(\bigsqcap P_k)^{ii}$ use only the signature Σ_0 it follows that $(\bigsqcap P_k)^\iota = (\bigsqcap P_k)^i \subseteq (\bigsqcap P_k)^{iii} = ((\bigsqcap P_k)^{ii})^\iota$. This shows that $(\bigsqcap P_k)^{ii}$ is a possible consequence of $\bigsqcap P_k$. Since D is minimal among the possible consequences of $\bigsqcap P_k$ we obtain $D \sqsubseteq (\bigsqcap P_k)^{ii}$. Thus $D \equiv (\bigsqcap P_k)^{ii}$.

Theorem 1 (Completeness). *Assume that Algorithm 2 terminates after the n-th iteration of the outer while loop. Then the set of GCIs* $\mathcal{B} = \{\bigsqcap P_k \sqsubseteq D_k \mid 0 \leq k \leq n\}$ *is complete for the background model i.*

Proof. We prove completeness by showing that Algorithm 2 finds exactly the same GCIs as Algorithm 1 initialised with the full background model i. This is done by induction. Let P_k, Π_k, M_k and S_k represent the outputs of Algorithm 1. Let $\mathbf{P}_k$, $\mathbf{\Pi}_k$, $\mathbf{M}_k$ and $\mathbf{S}_k$ represent the respective outputs of Algorithm 2. We prove by induction over k that

$$P_k = \mathbf{P}_k, \qquad \Pi_k = \mathbf{\Pi}_k, \qquad M_k = \mathbf{M}_k, \qquad S_k = \mathbf{S}_k \tag{2}$$

Base Case: The case $k = 0$ is trivial. *Step Case:* Assume that (2) holds for all $k < k_0$. *Part 1.* $\Pi_{k_0} = \mathbf{\Pi}_{k_0}$, $M_{k_0} = \mathbf{M}_{k_0}$, $S_{k_0} = \mathbf{S}_{k_0}$ follow immediately from the induction hypothesis and Lines 5-7 in Algorithm 1 and Lines 12-14 in Algorithm 2. *Part 2.* We show that $P_{k_0} = \mathbf{P}_{k_0}$. To do this, we only need to show that $\mathrm{pr}_{\mathbf{M}_{k_0}}(\mathbf{D}_l) = P_l''^{k_0}$ for all $1 \leq l < k_0$ (see Line 12 of Algorithm 1 and Line 19 of Algorithm 2). Lemma 4 shows that $P_l''^{k_0} = \mathrm{pr}_{M_{k_0}}\left((\bigsqcap P_l)^{ii}\right)$. By induction hypothesis and Part 1 we obtain $P_l''^{k_0} = \mathrm{pr}_{\mathbf{M}_{k_0}}\left((\bigsqcap \mathbf{P}_l)^{ii}\right)$. Then Lemma 5 proves that $\mathrm{pr}_{\mathbf{M}_{k_0}}(\mathbf{D}_l) = P_l''^{k_0}$ for all $1 \leq l < k_0$, and therefore $P_{k_0} = \mathbf{P}_{\mathbf{k_0}}$.

This finishes the induction proof. So we have shown that $\mathbf{P}_k = P_k$ for all $k \in \{1, \ldots, n\}$. Lemma 5 proves $\mathbf{D}_k = (\bigsqcap \mathbf{P}_k)^{ii} = (\bigsqcap P_k)^{ii}$. Hence the set of GCIs $\mathcal{B}$ that is found by Algorithm 2 is exactly the same as the set $\mathcal{B}_n$ that Algorithm 1 computes with the full background model i as input. Since Algorithm 1 is complete, Algorithm 2 must also be complete.

Termination, however, is more difficult. If the algorithm does not get stuck in the inner while-loop (Lines 6 to 10) then it is guaranteed to terminate. This is because outside the inner while loop it behaves just like Algorithm 1, and Algorithm 1 terminates. In summary, there remain two issues to be adressed: The existence and computation of minimal possible consequences and termination of the above algorithm.

5 Which Language Should be Used for the Knowledge Base?

So far we have not said anything about the description logic in which the knowledge base should be written. Algorithm 2 does not make any explicit requirements except that minimal possible consequences should exist. Of course, the whole algorithm only makes sense if it can terminate. In this section we try to find out for which logics this is the case.

The most natural choice for the logic of the knowledge base is $\mathcal{EL}_{\mathrm{gfp}}$. Unfortunately, $\mathcal{EL}_{\mathrm{gfp}}$ is not suitable, because it is not expressive enough to express that an individual is a counterexample to a given GCI $C \sqsubseteq D$. Intuitively, this is because $\mathcal{EL}_{\mathrm{gfp}}$ does not provide any form of negation. For example, it is impossible to state in $\mathcal{EL}_{\mathrm{gfp}}$ that `Henry` from the model from Figure 1 is not an instance of `Mother`. Therefore, it is impossible to state that `Henry` is a counter-example to the GCI `Father` $\sqsubseteq$ `Mother`. Because the expert cannot describe counter-examples the algorithm cannot terminate if $\mathcal{EL}_{\mathrm{gfp}}$ is used for the knowledge base.

5.1 $\mathcal{EL}_{\mathrm{gfp}}$ with Negated Concept Assertions

We have seen that we require at least some weak form of negation in the underlying knowledge base, or else the algorithm cannot terminate. On the other hand, we do not want to make the language of the knowledge base unnecessarily complicated. A simple extension are negated concept assertions.

A *negated concept assertion* is a statement of the form $\neg C(a)$, where C is a concept description. The semantics of negated concept assertions is defined in a straightforward way. Let $\mathcal{A}$ be an ABox that contains role assertions, concept assertions and negated concept assertions. An interpretation i is a model of $\mathcal{A}$ if and only if for all concept assertions $C(a)$ in $\mathcal{A}$ it holds that $a^i \in C^i$, and for all negated concept assertions $\neg C(a)$ it holds that $a^i \notin C^i$, and for all role assertions $r(a, b)$ it holds that $(a^i, b^i) \in r^i$.

In the setting that we consider in this subsection we are given a background model i. The concept assertions and negated concept assertions occurring in $\mathcal{A}$ shall use $\mathcal{EL}_{\mathrm{gfp}}$-concept descriptions over Σ_0, the signature of i. We do not explicitly use a TBox, but TBoxes are, of course, implicitly present within the $\mathcal{EL}_{\mathrm{gfp}}$-concept descriptions. Allowing (unfoldable) TBoxes explicitly would not result in more expressivity. Obviously in this setting counterexamples do exist and are easy to describe. To turn an individual a into a counterexample to a GCI $C \sqsubseteq D$ we simply need add $C(a)$ and $\neg D(a)$ to the ABox. However, there can still be situations (i. e. background models) where the algorithm cannot terminate.

Consider the background model i depicted in Figure 2. Let the signature be $\mathcal{N}_C = \{P, Q\}$ and $\mathcal{N}_r = \{r\}$. Assume that $\mathcal{A}$ is an ABox that has i as its model. Clearly, if $\mathcal{A}$ is empty then any interpretation is a model and thus any concept description D is a possible consequence of P. In particular this means that $\{x\}^i$ is not minimal among the possible consequences of P.

If $\mathcal{A}$ is not empty, then there is exactly one individual present in $\mathcal{A}$ because of the unique names assumption and because there is only one individual in the background model. We denote this individual by a. Let $\mathcal{A}$ contain the concept assertions $T_1(a), \ldots, T_t(a)$, and the negated concept assertions $\neg F_1(a), \ldots,$ $\neg F_f(a)$, and possibly a single role assertion $r(a, a)$. For every concept description F_k that occurs in a negated concept assertion we can define the Q-depth d_{F_k} of F_k. By d_{F_k} we denote the minimal role depth at which Q appears in F_k. Define $d = 1 + \max_{1 \leq k \leq f} d_{F_k}$.

Now, look at the model ι depicted in Figure 3. The model ι is obtained from i by attaching to x a sequence of nodes v_k, $k \in \{1, \ldots, d\}$ where each node is connected to its successor by the role r. The last of these new nodes v_d is in Q^ι. Clearly, the role assertion $r(a, a)$ holds in ι. All positive concept assertions from $\mathcal{A}$ hold in ι because they hold in i and i is a submodel of ι. All negated

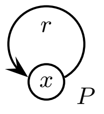

Fig. 2. The model i used in Section 5.1

concept assertions $N_k(a)$ from $\mathcal{A}$ also hold in ι, because Q occurs in N_k at a role depth less than d, but there is no path of length less than d leading from x to an individual in Q^ι. Therefore ι is a model of $\mathcal{A}$. It holds that $x \in E^\iota$, with $E = (A_E, \mathcal{T}_E)$ where $\mathcal{T}_E$ is defined as

$$\mathcal{T}_E = \{A_E \equiv P \sqcap \exists r.A_E \sqcap \underbrace{\exists r.\exists r \ldots \exists r}_{d \text{ times}}.Q\}.$$

Thus E is a possible consequence for P. In particular this proves that $\{x\}^i = P^{ii}$ is not a minimal possible consequence of P.

Now assume that the algorithm has reached a point where $P_k = \{P\}$. The condition required to leave the inner-while loop is that the expert accepts the GCI $P \sqsubseteq D$ where D is a minimal possible consequence for P. We have seen that this can only be the case if $D = P^{ii}$ (Lemma 5). But this can never happen, as for no ABox – be it empty or non-empty – P^{ii} is a minimal possible consequence of P. Hence, for our purposes negated concept assertions are an insufficient extension to $\mathcal{EL}$.

5.2 $\mathcal{EL}$ with $\bot$ and general TBoxes

The bottom concept $\bot$ is a concept constructor whose semantics is defined as $\bot^i = \emptyset$. We suggest to use $\mathcal{EL}$ with the bottom concept $\bot$ and general TBoxes (from now on denoted as $\mathcal{EL}_\bot$) as the DL for the knowledge base. First of all, this logic is a fragment of $\mathcal{EL}^{++}$, a well-supported, tractable DL that is used in many applications such as SNOMED.

$\mathcal{EL}_\bot$ is a minimal extension of $\mathcal{EL}$ in which it is possible to provide counterexamples. Consider for example the model from Figure 1. The model contains a counter-example to the GCI $\texttt{Father} \sqsubseteq \texttt{Mother}$, namely the individual `Henry`. To describe this counter-example in $\mathcal{EL}_\bot$ we have to express that `Henry` is not an instance of `Mother`. We can do this by extending the signature of the knowledge base by adding a new concept name $T_{\texttt{Henry}}$. Then we add the GCI $T_{\texttt{Henry}} \sqcap \texttt{Mother} \sqsubseteq \bot$ to the TBox and `Father(Henry)` and $T_{\texttt{Henry}}$`(Henry)` to the ABox. This implies that `Henry`, as an ABox-individual, must be an instance of `Father`, yet it cannot be an instance of `Mother`. Notice, that we need to extend the signature of the TBox, in order to describe the counterexample.

Termination is Possible Regarding termination of Algorithm 2 one can ask two questions. First, is it possible that an expert with an optimal strategy of providing counterexamples can force the algorithm to terminate? And second,

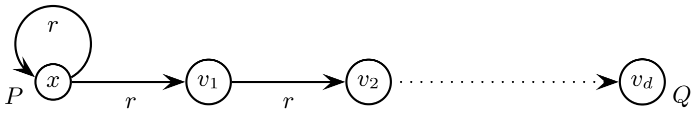

Fig. 3. The model ι used in Section 5.1

is it possible to modify the algorithm such that it terminates, even if the expert uses a suboptimal strategy? While the second question is part of ongoing work, we can give the answer to the first question for $\mathcal{EL}_\bot$.

We have seen in Section 4.2 that Algorithm 2 terminates if and only if it does not get stuck in the inner while loop (Lines 6 to 10). It will leave the inner while loop when $D \equiv (\bigsqcap P_k)^{ii}$ holds. That means, the expert can force the algorithm to terminate if she can come up with an ABox $\mathcal{A}_j$ and a TBox $\mathcal{T}_j$ such that $(\bigsqcap P_k)^{ii}$ is the only minimal possible consequence of $\bigsqcap P_k$.

Let i be the background model that is known to the expert and $\Sigma_0 = (\mathcal{N}_C, \mathcal{N}_r)$ its signature. We prove that enforcing termination is possible by providing a construction for such ABox and TBox from i and $\mathcal{A}_{j-1}$ and $\mathcal{T}_{j-1}$. For every $x \in \Delta_i$ extend the signature Σ_{j-1} by concept names T_x and F_x. Add an individual a_x for every $x \in \Delta_i$. $\mathcal{T}_j$ is obtained from $\mathcal{T}_{j-1}$ by adding statements

$$T_x \sqcap F_x \sqsubseteq \bot \qquad \text{for all } x \in \Delta_i, \tag{3}$$

$$A \sqsubseteq F_x \qquad \text{for all } A \in \mathcal{N}_C \text{ with x } \notin A^i, \tag{4}$$

$$\exists r. \bigsqcap \{F_y \mid y \in xr^i\} \sqsubseteq F_x \qquad \text{for all } r \in \mathcal{N}_r. \tag{5}$$

$\mathcal{A}_j$ is obtained from $\mathcal{A}_{j-1}$ by adding the following statements

$$A(a_x) \qquad \text{for every } x \in \Delta_i \text{ and for every } A \in \mathcal{N}_C \text{ with } x \in A^i, \tag{6}$$

$$r(a_x, a_y) \qquad \text{for every } r \in \mathcal{N}_r \text{ and for all } x, y \in \Delta_i \text{ with } (x, y) \in r^i, \tag{7}$$

$$T_x(a_x) \qquad \text{for every individual } x \in \Delta_i. \tag{8}$$

The concept T_x intuitively represents the properties that x does have while F_x represents the properties that x does not have. In the following we shall prove that for any arbitrary concept description C over the signature Σ_0 the concept C^{ii} is the only minimal possible consequence with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.

Lemma 6. *$\mathcal{A}_j$ and $\mathcal{T}_j$ are a representation of i.*

Proof. We have assumed that $\mathcal{A}_{j-1}$ and $\mathcal{T}_{j-1}$ are representations of the background model i. That means, there is a model ι of $\mathcal{A}_{j-1}$ and $\mathcal{T}_{j-1}$ such that ι restricted to Σ_0 is identical to i. We can further extend ι to a model $\bar{\iota}$ by defining $T_x^{\bar{\iota}} = \{x\}$ and $F_x^{\bar{\iota}} = \Delta_i \setminus \{x\}$. Then $\bar{\iota}$ is a model of $\mathcal{A}_j$ and $\mathcal{T}_j$ (it is simple to check that each of the statements (3) to (8) holds in $\bar{\iota}$). This shows that $\mathcal{A}_j$ and $\mathcal{T}_j$ are a representation of i.

Lemma 7. *Let C be any $\mathcal{EL}_{\mathrm{gfp}}$-concept description over the signature Σ_0. C^{ii} is a possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.*

Proof. We have already seen that $\bar{\iota}$ is a model of $\mathcal{A}_j$ and $\mathcal{T}_j$. Since C and C^{ii} use only the signature Σ_0 (and not the new concept names F_x and T_x, $x \in \Delta_i$) it holds that $C^i = C^{\bar{\iota}}$ and $C^{iii} = (C^{ii})^{\bar{\iota}}$. Lemma 2 states that $C^i = C^{iii}$ and thus $C^{\bar{\iota}} = (C^{ii})^{\bar{\iota}}$. Thus C^{ii} is a possible consequence of C in $\mathcal{A}_i$ and $\mathcal{T}_i$.

Lemma 8. *Let $C = (A_C, \mathcal{T}_C)$ be an $\mathcal{EL}_{\mathrm{gfp}}$-concept description over the signature Σ_0. Let $x \in \Delta_i$ be an individual. If there is a model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ such that $a_x^\iota \in C^\iota$ then $x \in C^i$ holds.*

Proof. Let ι be a model such that $a_x^\iota \in C^\iota$. From Lemma 1 it follows that there is a simulation $\zeta_{C\iota}$ from $\mathcal{T}_C$ to ι such that $(A_C, a_x^\iota) \in \zeta_{C\iota}$. Define ζ_{Ci} as follows:

$$\zeta_{Ci} = \{(B, y) \mid \exists z \in \Delta_\iota : z \notin F_y^\iota \text{ and } (B, z) \in \zeta_{C\iota}\}.$$

To prove that ζ_{Ci} is a simulation from $\mathcal{T}_C$ to i that contains (A_C, x) one must prove (S1), (S2) and $(A_C, x) \in \zeta_{Ci}$. Due to space restrictions we only show the interesting part (S2). Let $r \in \mathcal{N}_r$ be a role name, $(B, y) \in \zeta_{Ci}$ and $E \in Br_{\mathcal{T}_C}$. By definition of ζ_{Ci} there is some $z \in \Delta_\iota$ such that $z \notin F_y^\iota$ and $(B, z) \in \zeta_{C\iota}$. Because $\zeta_{C\iota}$ is a simulation there must be some $\bar{z} \in zr^i$ such that $(E, \bar{z}) \in \zeta_{C\iota}$.

Suppose that for all $\bar{y} \in yr^i$ it holds that $(E, \bar{y}) \notin \zeta_{Ci}$. This implies that $\bar{z} \in F_{\bar{y}}^\iota$ for all $\bar{y} \in yr^i$. But $\mathcal{T}_j$ contains the statement $\exists r. \bigsqcap\{F_{\bar{y}} \mid \bar{y} \in yr^i\} \sqsubseteq F_y$. The above proves $\bar{z} \in (\bigsqcap\{F_{\bar{y}} \mid \bar{y} \in yr^i\})^\iota$ and thus $z \in (\exists r. \bigsqcap\{F_{\bar{y}} \mid \bar{y} \in yr^i\})^\iota \subseteq F_y^\iota$. This contradicts $z \notin F_y^\iota$. We have shown by contradiction that (S2) holds. Together with the omitted steps this proves that ζ_{Ci} is a simulation from $\mathcal{T}_C$ to i such that $(C, x) \in \zeta_{Ci}$. Lemma 1 shows that $x \in C^i$.

Lemma 9. *Let $C = (A_C, \mathcal{T}_C)$ be an $\mathcal{EL}_{\text{gfp}}$-concept description over the signature Σ_0. Let $x \in \Delta_i$. If $x \in C^i$ holds then $\mathcal{A}_j, \mathcal{T}_j \models C(a_x)$*

Proof. We need to show that for any model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ it holds that $a_x^\iota \in C^\iota$. Because of $x \in C^i$ there must be a simulation ζ_{Ci} from $\mathcal{T}_C$ to i containing (A_C, x). Define $\zeta_{C\iota} = \{(B, a_y^\iota) \mid (B, y) \in \zeta_{Ci}\}$. As above, we omit the proofs for (S1) and $(A_C, a_x^\iota) \in \zeta_{C\iota}$. We only prove the interesting step (S2). Let $r \in \mathcal{N}_r$ be a role name, let $(B, a_y^\iota) \in \zeta_{C\iota}$, and let $\bar{B} \in Br_{\mathcal{T}_C}$. $(B, a_y^\iota) \in \zeta_{C\iota}$ implies $(B, y) \in \zeta_{Ci}$. Because ζ_{Ci} is a simulation there is some $\bar{y} \in yr^i$ such that $(\bar{B}, \bar{y}) \in \zeta_{Ci}$. The latter implies that $(\bar{B}, a_{\bar{y}}^\iota) \in \zeta_{C\iota}$. $\bar{y} \in yr^i$ implies that $\mathcal{A}_j$ contains a statement $r(a_y, a_{\bar{y}})$ and therefore $a_{\bar{y}}^\iota \in a_y^\iota r^\iota$. This proves (S2). From Lemma 1 and the fact that $\zeta_{C\iota}$ is a simulation it then follows that $a_x^\iota \in C^\iota$. Since ι was an arbitrary model it follows that $\mathcal{A}_j, \mathcal{T}_j \models C(a_x)$.

Theorem 2. *Let C be any $\mathcal{EL}_{\text{gfp}}$-concept description. C^{ii} is the only minimal possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.*

Proof. Lemma 7 proves that C^{ii} is a possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$. Let D be another possible consequence of C. That means there is a model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ such that $C^\iota \subseteq D^\iota$. From Lemma 9 it follows that $a_x^\iota \in C^\iota$ for all $x \in C^i$ and thus also $a_x^\iota \in D^\iota$. But then Lemma 8 implies that $x \in D^i$ for all $x \in C^i$, i. e. $C^i \subseteq D^i$. Because most specific concepts are minimal with respect to $\sqsubseteq$ we obtain that $C^{ii} \sqsubseteq D$. This proves that C^{ii} is the the only minimal possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.

6 Summary and Open Questions

We have shown that the model exploration algorithm from [6] can be adapted to a new setting. In this new setting, the counter-examples for the rejected GCIs are stored in an ABox instead of a model.

In order to adapt the algorithm we have introduced the notion of minimal possible consequences. We have seen that minimal possible consequences can be used to replace model-based most specific concepts in our new setting. We have presented an exploration algorithm (Algorithm 2) and proved its completeness. We have pointed out that not all DL languages are suitable to describe counter-examples. It is crucial that this language provides negation or disjointness.

This is ongoing work, and the questions of termination and the existence of minimal possible consequences remain open. Termination has partly been addressed in this paper. We have seen that an expert with an optimal strategy can force the algorithm to terminate if $\mathcal{EL}_\bot$ is used for the knowledge base. However, termination is not guaranteed if the expert uses a different strategy.

Existence of minimal possible consequences, has not been addressed in this work. The author is currently working on a modified tableau algorithm which might be used to compute minimal possible consequences for $\mathcal{EL}_\bot$.

References

1. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: *Proc. of IJCAI 2003*, pp. 319–324. Morgan Kaufmann, San Francisco (2003).
2. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: *Proc. of IJCAI 2003*, pp. 319–324. Morgan Kaufmann, San Francisco (2003).
3. Baader F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press (2003).
4. Baader, F. and Distel, F.: Exploring finite models in the description logic $\mathcal{EL}_{\mathrm{gfp}}$. LTCS-Report 08-05, Chair for Automata Theory, TU Dresden (2008).
5. Baader, F. and Distel, F.: A finite basis for the set of $\mathcal{EL}$-implications holding in a finite model. In: *Proc. of ICFCA '08*, LNAI, vol. 4933, pp. 46–61. Springer (2008).
6. Baader, F. and Distel, F.: Exploring finite models in the description logic $\mathcal{EL}_{\mathrm{gfp}}$. In: *Proc. of ICFCA 2009*. Springer (2009).
7. Baader, F., Ganter, B., Sattler, U., and Sertkaya, B.. Completing description logic knowledge bases using formal concept analysis. In: *Proc. of the IJCAI 2007*. AAAI Press/The MIT Press (2007).
8. Baader, F., Lutz, C., and Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: *Proc. of IJCAR 2006*, LNAI, vol. 4130, pp. 287–291. Springer (2006).
9. Ganter, B. and Wille, R.: *Formal Concept Analysis: Mathematical Foundations.* Springer, New York (1997).
10. Horrocks, I., Patel-Schneider, P., and van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26 (2003).
11. Rudolph, S.: Exploring relational structures via FLE. In: *ICCS*, LNCS, vol. 3127, pp. 196–212 (2004).
12. Spackman, K.A., Campbell, K.E., and Cote, R.A.: SNOMED RT: A reference terminology for health care. *J. of the American Medical Informatics Association*, pp. 640–644 (1997). Fall Symposium Supplement.
13. The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29 (2000).