DeFind: A Protege Plugin for Computing Concept Definitions in $\mathcal{EL}$ Ontologies[*]

Denis Ponomaryov[1,2] and Stepan Yakovenko[2]

[1] Institute of Mathematics, Novosibirsk State University
[2] Institute of Informatics Systems, Novosibirsk State University
`ponom@iis.nsk.su, stiv.yakovenko@gmail.com`

Abstract. We introduce an extension to the Protégé ontology editor, which allows for discovering concept definitions, which are not explicitly present in axioms, but are logically implied by an ontology. The plugin supports ontologies formulated in the Description Logic $\mathcal{EL}$, which underpins the OWL 2 EL profile of the Web Ontology Language and despite its limited expressiveness captures most of the biomedical ontologies published on the Web. The developed tool allows to verify whether a concept can be defined using a vocabulary of interest specified by a user. In particular, it allows to decide whether some vocabulary items can be omitted in a formulation of a complex concept. The corresponding definitions are presented to the user and are provided with explanations generated by an ontology reasoner.

1   Introduction

The development and application of large terminological systems pose new challenges for ontology engineering and automated reasoning tools. A question which becomes evidently important in the context of large ontologies is whether their content and logical consequences are easy to comprehend. To address this problem, a number of visualization and explanation tools has been integrated into ontology editing environments such as, e.g., visualization and explanation services implemented for the Protégé ontology editor. It is common that an expert exploring an ontology encounters concepts, which are formulated in a vocabulary she is not familiar with. More generally, only a certain part of an ontology vocabulary may be familiar to an expert, while the remaining part may be not. When a formulation of some concept employs vocabulary items unknown to an expert, typically she would like to know whether this concept can be reformulated in the familiar vocabulary. To give a simplified example, suppose an ontology of cuisines contains axiom $Dumplings \sqcap Entree \sqsubseteq Gnocci$ (stating that $Dumplings$ being $Entree$ are $Gnocci$) together with concept inclusions $Gnocci \sqsubseteq Dumplings$ and $Dumplings \sqsubseteq Entree$. Assume an expert is familiar with concepts $Entree$ and $Gnocci$ and she encounters the concept $Dumplings \sqcap Entree$ mentioned in the ontology. First, one may notice that this

concept can be simplified, i.e., it is equivalent to *Dumplings* wrt the ontology, due to the inclusions above. Second, the ontology entails that *Dumplings* and *Gnocci* are equivalent, thus, the original concept can be reformulated as *Gnocci* (or as $Gnocci \sqcap Entree$) in the vocabulary known to the user.

In this paper, we introduce DeFind, an extension to the Protégé ontology editor, which allows to find concept definitions in a user specified vocabulary consisting of concept and relations names from an input ontology. In particular, it allows to verify whether some concept or relation names can be omitted in a formulation of a (complex) concept. To the best of our knowledge, this is the first implementation of these reasoning services as a part of an ontology editing tool. Specifically, for a given ontology $\mathcal{O}$, vocabulary Σ, and a concept C of interest, DeFind computes definitions of C wrt $\mathcal{O}$ in Σ, i.e. concepts $\{D_1, \ldots, D_n\}$ (whenever they exist) such that D_i contains symbols only from Σ and $\mathcal{O} \models C \equiv D_i$, for all $i = 1, \ldots, n$. DeFind supports ontologies formulated in the Description Logic $\mathcal{EL}$, which allows for building concepts using conjunction and existential restriction, and includes built-in concepts such as $\bot$ and $\top$. For example, it is possible to state in $\mathcal{EL}$ that some concepts are disjoint (e.g., $Entree \sqcap Dessert \sqsubseteq \bot$), specify subsumption relationship between concepts ($Dumplings \sqcap Entree \sqsubseteq Gnocci$) or domains of relations ($\exists hasIngredient.\top \sqsubseteq Food$), and use existential restriction to specify relationships of other kinds (e.g., $Salad \sqsubseteq \exists hasDressing.\top$, $\exists hasIngredient.Meat \sqsubseteq NonVegeterianFood$). The expressive features of $\mathcal{EL}$ although limited (e.g., negation/disjunction of concepts is not allowed), are sufficient to capture a great variety of ontologies. Most of the biomedical ontologies published on the Web fall within the formalism of $\mathcal{EL}$, which underpins the OWL 2 EL profile of the Web Ontology Language.

2 Techniques

In the Description Logic $\mathcal{EL}$, concepts are built using a countably infinite alphabet of *roles names* $\mathsf{N_r}$ and *concept names* $\mathsf{N_c}$, with two distinguished concepts $\bot, \top \in \mathsf{N_c}$. The notion of *concept* is defined inductively: any element of $\mathsf{N_c}$ is a concept and if C, D are concepts and $r \in \mathsf{N_r}$ is a role then $C \sqcap D$ and $\exists r.C$ are concepts. A *concept inclusion* is an expression of the form $C \sqsubseteq D$, where C, D are concepts. A *concept equivalence* is a expression $C \equiv D$. An *ontology* is a finite set of concept inclusions and equivalences (called *axioms*). A *signature* is a subset of $\mathsf{N_r} \cup \mathsf{N_c} \setminus \{\bot, \top\}$. The signature of a concept C, denoted as $\mathtt{sig}\,(C)$, is the set of role and concept names (excluding $\bot$ and $\top$), which occur in C. The signature of a concept inclusion or ontology is defined similarly. Semantics is defined by using the notion of interpretation, which is a pair $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$, where Δ is a universe and $\cdot^{\mathcal{I}}$ is a function which maps every concept name from $\mathsf{N_c}$ to a subset of Δ such that $(\bot)^{\mathcal{I}} = \varnothing$, $(\top)^{\mathcal{I}} = \Delta$, and every role name to a subset of $\Delta \times \Delta$. This function is extended to arbitrary concepts by setting $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ and $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta \mid \exists y \, \langle x, y \rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$. An interpretation $\mathcal{I}$ is a model of a concept inclusion $C \sqsubseteq D$ (written as $\mathcal{I} \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Similarly, $\mathcal{I} \models C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. An interpretation is a model of an ontology $\mathcal{O}$ if it is a model of every axiom of $\mathcal{O}$. An ontology $\mathcal{O}$ *entails* a concept inclusion $C \sqsubseteq D$ (written as $\mathcal{O} \models C \sqsubseteq D$) if $\mathcal{I} \models \mathcal{O}$ yields $\mathcal{I} \models C \sqsubseteq D$, for any interpretation $\mathcal{I}$.

Let Σ be a signature. We say that a concept C is Σ-*definable* wrt an ontology $\mathcal{O}$ if $\mathcal{O} \models C \equiv D$, where $\mathtt{sig}\,(D) \subseteq \Sigma$. The concept D is called Σ-*definition* of C (wrt ontology $\mathcal{O}$). For example, the concept $Dumplings \sqcap Entree$ is Σ-definable wrt the ontology from the introduction, where $\Sigma = \{Gnocci\}$. It is known that in $\mathcal{EL}$ a concept C is Σ-definable wrt an ontology $\mathcal{O}$ iff it holds $\mathcal{O} \cup \mathcal{O}^* \models C \sqsubseteq C^*$, where $\mathcal{O}^*$ and C^* are "copies" of $\mathcal{O}$ and C, respectively, obtained by an injective renaming of all non-Σ-symbols into "fresh" ones, not occurring in $\mathcal{O}$ and C. Indeed, if $\mathcal{O} \models C \equiv D$, for some concept D, with $\mathtt{sig}\,(D) \subseteq \Sigma$, then it holds $\mathcal{O} \models C \sqsubseteq D$ and $\mathcal{O}^* \models D \sqsubseteq C^*$, which means $\mathcal{O} \cup \mathcal{O}^* \models C \sqsubseteq C^*$. The converse can be proved constructively by computing the corresponding concept D from a proof of the inclusion $C \sqsubseteq C^*$ from $\mathcal{O} \cup \mathcal{O}^*$; we provide a justification below. Various proof systems have been proposed for the Description Logic $\mathcal{EL}$ in the literature. For example, it follows from the results in [1] that the following set of inference rules is sound and complete for entailment of concept inclusions $C \sqsubseteq D$ from an ontology $\mathcal{O}$, where C and D occur in the axioms of $\mathcal{O}$:

$$\mathsf{R_0}\ \frac{}{C \sqsubseteq C} \qquad \mathsf{R_\top}\ \frac{}{C \sqsubseteq \top} \qquad \mathsf{R_\bot}\ \frac{}{\bot \sqsubseteq C} \qquad \mathsf{R_\sqsubseteq}\ \frac{C \sqsubseteq E}{C \sqsubseteq F}\ E \bowtie F \in \mathcal{O},\ \bowtie\, \in \{\sqsubseteq, \equiv\}$$

$$\mathsf{R_\sqcap^-}\ \frac{C \sqsubseteq E_1 \sqcap E_2}{C \sqsubseteq E_1\ \ C \sqsubseteq E_2} \qquad \mathsf{R_\sqcap^+}\ \frac{C \sqsubseteq E_1\ \ C \sqsubseteq E_2}{C \sqsubseteq E_1 \sqcap E_2}\ E_1 \sqcap E_2 \text{ occurs in } \mathcal{O}$$

$$\mathsf{R_\exists^\bot}\ \frac{C \sqsubseteq \exists r.E\ \ E \sqsubseteq \bot}{C \sqsubseteq \bot} \qquad \mathsf{R_\exists}\ \frac{C \sqsubseteq \exists r.E\ \ E \sqsubseteq F}{C \sqsubseteq \exists r.F}\ \exists r.F \text{ occurs in } \mathcal{O}$$

Fig. 1. Basic inference rules for reasoning in $\mathcal{EL}$

An *inference* is a triple $\langle$rule name, premises, conclusion$\rangle$ of the form $\langle \mathsf{R}, \{\varphi_1, .., \varphi_n\}, \psi \rangle$, where $n \geqslant 0$ and ψ is obtained by an inference rule R with premises $\varphi_1,, \ldots, \varphi_n$. A *proof* of a concept inclusion ψ (from an ontology $\mathcal{O}$) is a set of inferences $\{\iota_1, \ldots, \iota_n\}$, where $n \geqslant 1$, such that ψ is the conclusion of ι_n and for all $k = 1, \ldots, n$ and any axiom φ, if φ is a premise of ι_k, then there is a unique $j < k$ such that φ is the conclusion of ι_j and if φ is the conclusion of ι_k and $k \neq n$ then there is $l > k$ such that φ is a premise of ι_l. Observe that due to the injective renaming of non Σ-symbols it holds $\mathtt{sig}\,(\mathcal{O}) \cap \mathtt{sig}\,(\mathcal{O}^*) \subseteq \Sigma$ and whenever a concept C occurs in ontology $\mathcal{O}$, we have $\mathtt{sig}\,(C) \subseteq \mathtt{sig}\,(\mathcal{O})$ and hence, $\mathtt{sig}\,(C^*) \subseteq \mathtt{sig}\,(\mathcal{O}^*)$. If there is a concept D such that $\mathtt{sig}\,(D) \subseteq \Sigma$ and $\mathcal{O} \cup \mathcal{O}^* \models \{C \sqsubseteq D,\ D \sqsubseteq C^*\}$ (in which case D is called *interpolant* for $C \sqsubseteq C^*$) then due to the renaming it holds that $\mathcal{O} \models \{C \sqsubseteq D, D \sqsubseteq C\}$ and thus, $\mathcal{O} \models C \equiv D$, i.e., D is a definition of concept C in signature Σ.

Theorem 1. *Let Σ be a signature, $\mathcal{O}_1, \mathcal{O}_2$ ontologies and C_1, C_2 concepts such that $\mathtt{sig}\,(\mathcal{O}_1) \cap \mathtt{sig}\,(\mathcal{O}_2) \subseteq \Sigma$ and C_i occurs in $\mathcal{O}_i$, for $i = 1, 2$. If $\mathcal{O}_1 \cup \mathcal{O}_2 \models C_1 \sqsubseteq C_2$ then there exists a concept D such that $\mathtt{sig}\,(D) \subseteq \Sigma$ and $\mathcal{O}_1 \cup \mathcal{O}_2 \models \{C_1 \sqsubseteq D,\ D \sqsubseteq C_2\}$.*

Proof. We note the following property (referred to as $\star$), which immediately follows from the definition of the inference rules: if there is a proof of a concept inclusion $C \sqsubseteq E$ from an ontology $\mathcal{O}$ then either it is obtained by one of the rules $\mathsf{R_\top}, \mathsf{R_\bot}, \mathsf{R_\exists^\bot}$, or E occurs in C or $\mathcal{O}$. We use induction on the number of inferences

in a proof $\langle \iota_i, \ldots, \iota_n \rangle$ of a concept inclusion $\varphi = C_1 \sqsubseteq C_2$ from $\mathcal{O}_1 \cup \mathcal{O}_2$. For $n = 1$, if φ is obtained by R_0 or $\mathsf{R}_\top$ then $\mathtt{sig}\,(C_2) \subseteq \Sigma$ and thus, C_2 is an interpolant for φ. If it is obtained by $\mathsf{R}_\bot$, then $\bot$ is an interpolant. For the induction step, if φ is obtained by $\mathsf{R}_\exists^\bot$ then $\bot$ is an interpolant. If the rule is $\mathsf{R}_\sqcap^+$ then by the induction assumption there is an interpolant D_i for each premise $C \sqsubseteq E_i$, $i = 1, 2$, and hence, $D_1 \sqcap D_2$ is an interpolant for φ. If the rule is $\mathsf{R}_\sqsubseteq$ then $E \bowtie F \in \mathcal{O}_1 \cup \mathcal{O}_2$, for some $i = 1, 2$, where $\bowtie \in \{\sqsubseteq, \equiv\}$. If $i = 1$ then it follows from $\mathtt{sig}\,(F) \subseteq \mathtt{sig}\,(\mathcal{O}_2)$ that $\mathtt{sig}\,(F) \subseteq \Sigma$ and thus, F is an interpolant for φ. If $i = 2$ then $\mathtt{sig}\,(E) \subseteq \mathtt{sig}\,(\mathcal{O}_2)$, thus by the induction assumption there is an interpolant for $C \sqsubseteq E$, which is an interpolant for φ. If the rule is $\mathsf{R}_\sqcap^-$, consider its premise $C \sqsubseteq E_1 \sqcap E_2$; w.l.o.g we assume that $\varphi = C \sqsubseteq E_1$. It follows from $(\star)$ that $\mathtt{sig}\,(E_1 \sqcap E_2) \subseteq \mathtt{sig}\,(\mathcal{O}_i)$, for some $i = 1, 2$. If $i = 2$, then there is an interpolant for $C \sqsubseteq E_1 \sqcap E_2$, which is an interpolant for φ. If $i = 1$ then $\mathtt{sig}\,(E_1) \subseteq \mathtt{sig}\,(\mathcal{O}_2)$ yields $\mathtt{sig}\,(E_1) \subseteq \Sigma$ and hence, E_1 is an interpolant for φ. Finally, if the rule is $\mathsf{R}_\exists$ then it follows from $(\star)$ that $\mathtt{sig}\,(\exists r.E) \subseteq \mathtt{sig}\,(\mathcal{O}_i)$, for some $i = 1, 2$. If $i = 2$ then there is an interpolant for $C \sqsubseteq \exists r.E$, which is an interpolant for φ. If $i = 1$ then it follows from the condition $\mathtt{sig}\,(\exists r.F) \subseteq \mathtt{sig}\,(\mathcal{O}_2)$ that $r \in \Sigma$ and by the induction assumption there is an interpolant D for $E \sqsubseteq F$. Then $\exists r.D$ is an interpolant for φ. $\square$

The proof of the theorem gives an idea of a recursive algorithm for computing interpolants and hence, concept definitions, by traversing proofs of $C \sqsubseteq C^*$ from the union $\mathcal{O} \sqcup \mathcal{O}^*$. In general, there can exist several proofs and each of them can yield a different definition of the concept C. Although the idea is simple, its implementation is not straightforward. First, it requires an ontology reasoner, which is not only able to decide entailment of concept inclusions from an ontology, but supports proof tracing, i.e., provides proofs as certificates for entailment. Second, the method presented by Theorem 1 has to rely on a proof system implemented by a reasoner. DeFind employs ELK [1], which is a highly optimized reasoner for the Description Logic $\mathcal{EL}$ and its extensions. The add-ons to ELK, which provide proof tracing and explanation services, employ the set of inference rules given in Figure 2. Showing a direct analogue of Theorem 1 for this proof system is not straightforward, since (in contrast to the rules from Figure 1) there exist proofs, from which an interpolant can not be directly computed. However, it can be shown that there always exists (at least a single) proof, which is appropriate for computing interpolants and hence, concept definitions.

DeFind implements a recursive procedure, which traverses proofs obtained by using the tracing service for ELK reasoner. Proofs are built for the entailment $\mathcal{O} \cup \mathcal{O}^* \models C \sqsubseteq C^*$, where $\mathcal{O}^*$ and C^* are "copies" (of a given ontology $\mathcal{O}$ and concept C) constructed wrt a specified signature Σ. The procedure applies recursively the rules below to compute a label expression $\mathsf{L}(\varphi)$ for every concept inclusion φ appearing as the conclusion of an inference and outputs a label computed for $C \sqsubseteq C^*$. Whenever there are several proofs for the same φ, each of them is traversed. A *label* is a concept formulated in an extension of $\mathcal{EL}$ with disjunction $\sqcup$ (denoted as $\mathcal{EL}_\sqcup$) and a distinguished "empty" concept ϵ, for which the following holds: $\epsilon \sqcap D = \epsilon$, $\epsilon \sqcup D = D$, $\exists r.\epsilon = \epsilon$, and $\exists r.(D \sqcup E) = \exists r.D \sqcup \exists r.E$, for any concepts D, E and role r. By using the latter equation, the notion of Disjunctive Normal Form of a $\mathcal{EL}_\sqcup$-concept is naturally defined. We say that a

$$\mathsf{S}_0 \; \frac{}{C \sqsubseteq C} \qquad \mathsf{S}_\top \; \frac{}{C \sqsubseteq \top} \qquad \mathsf{S}_\bot \; \frac{}{\bot \sqsubseteq C} \qquad \mathsf{S}_{ax} \; \frac{}{C \sqsubseteq E} \; C \sqsubseteq E \in \mathcal{O}$$

$$\mathsf{S}_\sqsubseteq \; \frac{C_0 \sqsubseteq C_1 \ldots C_{n-1} \sqsubseteq C_n}{C_0 \sqsubseteq C_n} \qquad \mathsf{S}_\equiv \; \frac{}{C_j \sqsubseteq C_k} \; C_i \equiv C_{i+1} \in \mathcal{O}, \; 0 \leqslant i < n, \; 0 \leqslant j,k \leqslant n$$

$$\mathsf{S}_\sqcap^- \; \frac{}{C_1 \sqcap \ldots \sqcap C_n \sqsubseteq C_i} \qquad \mathsf{S}_\sqcap^+ \; \frac{C \sqsubseteq E_1 \; \ldots \; C \sqsubseteq E_n}{C \sqsubseteq E_1 \sqcap \ldots \sqcap E_n}$$

$$S_\exists^\bot \; \frac{}{\exists r.\bot \sqsubseteq \bot} \qquad \mathsf{S}_\exists \; \frac{C \sqsubseteq E}{\exists r.C \sqsubseteq \exists r.E}$$

Fig. 2. Inference rules used by the proof tracing service for ELK reasoner

label expression is *empty* if it equals ϵ. Initially the label of every conclusion appearing in a proof is assumed to be empty. Each rule below is provided with a name and gives a label to the conclusion φ of an inference being visited during proof traversal, depending on the type of the inference rule and the labels of its premises. We use the notations from Figure 2 for the premises of each inference rule mentioned below:

(L$_\bot$) rule is $\mathsf{S}_\bot \Rightarrow \mathsf{L}(\varphi) := \mathsf{L}(\varphi) \sqcup \bot$
(L$_\Sigma$) rule is one of $\mathsf{S}_0, \mathsf{S}_\top, \mathsf{S}_{ax}, \mathsf{S}_\equiv, \mathsf{S}_\sqcap^-, \mathsf{S}_\exists^\bot$, $\varphi = C \sqsubseteq E$, and $\mathsf{sig}\,(E) \subseteq \Sigma \Rightarrow$
    $\mathsf{L}(\varphi) := \mathsf{L}(\varphi) \sqcup E$
(L$_\sqsubseteq$) rule is $\mathsf{S}_\sqsubseteq \Rightarrow \mathsf{L}(\varphi) := \mathsf{L}(\varphi) \sqcup \bigsqcup_{i=1,\ldots,n} \mathsf{L}(C_{i-1} \sqsubseteq C_i)$
(L$_\sqcap^+$) rule is $\mathsf{S}_\sqcap^+ \Rightarrow \mathsf{L}(\varphi) := \mathsf{L}(\varphi) \sqcup \prod_{i=1,\ldots,n} \mathsf{L}(C \sqsubseteq E_i)$
(L$_\exists$) rule is $\mathsf{S}_\exists$ and $r \in \Sigma \Rightarrow \mathsf{L}(\varphi) := \mathsf{L}(\varphi) \sqcup \exists r.\mathsf{L}(C \sqsubseteq E)$

Theorem 2. *Let Σ be a signature, $\mathcal{O}$ ontology, and C a concept occurring in $\mathcal{O}$. The concept C is Σ-definable wrt $\mathcal{O}$ iff the procedure returns a non-empty $\mathcal{EL}_\sqcup$-expression $\mathsf{L}(C \sqsubseteq C^*)$. Every conjunct from the Disjunctive Normal Form of $\mathsf{L}(C \sqsubseteq C^*)$ is a $\mathcal{EL}$-concept and it is a Σ-definition of C wrt $\mathcal{O}$.*

3 Complexity of Definitions

We now estimate the size and the number of definitions computed by DeFind. The above mentioned procedure recursively computes labels for conclusions of inferences appearing in proofs. Under the assumption that every proof is computed in polynomial time in the size of an input ontology $\mathcal{O}$ (which is true when using the ELK reasoner), the size of every proof is polynomially bounded by the size of $\mathcal{O}$. For any conclusion, the label is computed once, when traversing a single proof. The rules (L$_\bot$), (L$_\Sigma$) give a label concept from a conclusion of some inference and hence, its size is bounded by the size of $\mathcal{O}$. Every other rule of the procedure gives a label, which is obtained as a combination (under $\sqcup$, $\sqcap$, or $\exists r.$, with $r \in \mathsf{sig}\,(\mathcal{O})$) of k expressions computed at the previous steps of the recursion, where $k \geqslant 1$ is the number of premises of an inference rule and hence, k is bounded by the size of the proof. In particular, the rules $\mathsf{L}_\sqsubseteq$, $\mathsf{L}_\sqcap^+$ give a label, which is at most k times longer than the labels of the premises of $\mathsf{S}_\sqsubseteq$, $\mathsf{S}_\sqcap^+$. Therefore, the size of any label computed from a single proof is at
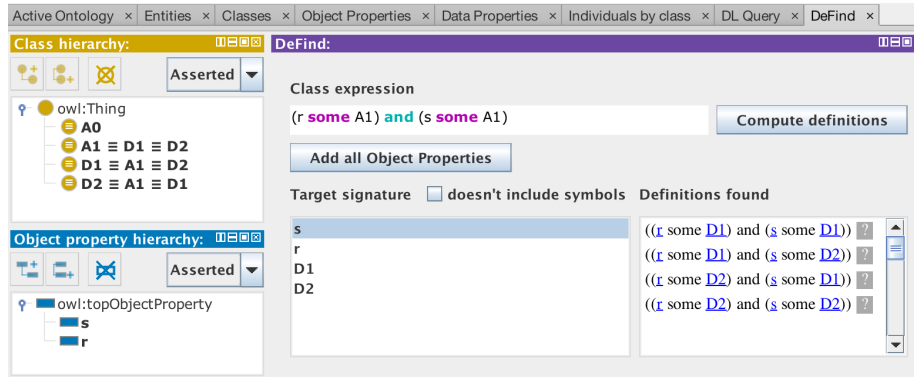
most exponential in the size of the ontology. Since every proof is polynomially bounded, there are at most exponentially many proofs for any conclusion and hence, the output of the procedure has at most exponential size. By Theorem 2, every conjunct from the DNF of the output label is a definition and therefore, the upper bound on the number of definitions computed by DeFind is double exponential in the size of the ontology. To estimate the size of the computed definitions one can w.l.o.g. assume that every rule of the procedure gives a label in DNF and show the following by induction. Every conjunct from a label obtained by some rule of the procedure is either a concept from a conclusion of an inference from a proof, or a combination (under $\sqcup$, $\sqcap$, or $\exists r.$) of conjuncts from the labels computed at the previous steps. Therefore, the size of every definition is at most exponential in the size of the input ontology. We now show that these bound are tight by giving an example of an ontology $\mathcal{O}$, concept C, and a signature Σ such that there is a double exponential number of shortest Σ-definitions of C wrt $\mathcal{O}$, where every definition has size exponential in the size of $\mathcal{O}$. Let $\Sigma = \{r, s, D_1, D_2\}$, $C = A_0$, and for $n \geqslant 1$, let $\mathcal{O}$ consist of axioms: $A_0 \equiv \exists r.A_1 \sqcap \exists s.A_1, \ldots, A_{n-1} \equiv \exists r.A_n \sqcap \exists s.A_n$, $A_n \equiv D_1$, $A_n \equiv D_2$. Observe that $\mathcal{O} \models A_n \equiv D_1 \sqcup D_2$, hence, $\mathcal{O} \models A_{n-1} \equiv \exists r.(D_1 \sqcup D_2) \sqcap \exists s.(D_1 \sqcup D_2)$. Converting this expression into DNF gives four Σ-definitions of A_{n-1} wrt $\mathcal{O}$, i.e., concepts $\exists r.D_i \sqcap \exists s.D_j$, where $i, j = 1, 2$. Similarly, A_0 is equivalent wrt $\mathcal{O}$ to a $\mathcal{EL}_{\sqcup}$-concept, which has 2^n occurrences of $D_1 \sqcup D_2$. Converting this concept into DNF gives a double exponential number of Σ-definitions of A_0 wrt $\mathcal{O}$ and it can be shown that no shorter Σ-definitions exist.

4 Features of DeFind

DeFind (available at https://github.com/stiv-yakovenko/defind) requires the ELK reasoner [1] and proof explanation plugin [2] for Protégé. The interface of DeFind is given in Figure 4. The screenshot shows Σ-definitions of concept $\exists r.A_1 \sqcap \exists s.A_1$ computed by Defind wrt ontology $\mathcal{O}$ and signature Σ from the previous section for $n = 1$. The user can input a complex concept in the Manchester OWL Syntax into the field "Class expression" or drag-and-drop a concept name from the class hierarchy of the ontology. To specify concept and role names for the target signature, one can drag-and-drop items from the class or object property hierarchy. One can simultaneously add all role names from the ontology into the target signature by pressing "Add all Object Properties" button. Selecting "doesn't include symbols" option sets the target signature to consist of all those concept an role names from the ontology, which are not listed in the "Target signature" field. On pressing "Compute definitions", the corresponding Σ-definitions are computed (or a notification is shown if no definition exists). On pressing the question mark at the right-hand side of a definition, an explanation is shown, why it corresponds to the specified class expression: the Protégé proof explanation plugin is called to visualize a proof (see, e.g., Figure 5 in [2]) of the equivalence of the class expression to the concept computed by DeFind.

5 Related Work

In [3], a similar approach is implemented to compute concept definitions from tableaux proofs in the Description Logic $\mathcal{ALC}$ and its extensions. The authors

Active Ontology × | Entities × | Classes × | Object Properties × | Data Properties × | Individuals by class × | DL Query × | DeFind ×

Class hierarchy:

Asserted ▼

- owl:Thing
  - A0
  - A1 ≡ D1 ≡ D2
  - D1 ≡ A1 ≡ D2
  - D2 ≡ A1 ≡ D1

Object property hierarchy:

Asserted ▼

- owl:topObjectProperty
  - s
  - r

DeFind:

Class expression

(r **some** A1) **and** (s **some** A1)

Compute definitions

Add all Object Properties

Target signature ☐ doesn't include symbols   **Definitions found**

s	((r some D1) and (s some D1))
r	((r some D1) and (s some D2))
D1	((r some D2) and (s some D1))
D2	((r some D2) and (s some D2))

Fig. 3. Interface of DeFind in the Protégé ontology editor

provide a constructive proof of an analogue of Theorem 1 and derive an algorithm, which computes a definition of at most double exponential size, whenever there exists one in a target signature. This bound is tight and is one exponential larger than the size of the shortest definitions in $\mathcal{EL}$. It is known that general tableaux methods, when applied to $\mathcal{EL}$, provide exponentially longer proofs than the ones obtained in the proof systems from this paper. It should be noted that the algorithm from [3] computes a single definition from a proof, whereas DeFind computes several definitions, the number of which is bounded by a double exponential in the size of the ontology. DeFind explores all proofs provided by a reasoner and traversing a single proof can potentially give several definitions for a concept of interest. In [4], the authors propose an algorithm, which finds a single concept definition in a signature of interest wrt a normalised $\mathcal{EL}$-ontology. The algorithm computes an algebraic representation of an ontology as a canonical model provided with an information on how each element of the model is obtained. Essentially, it implements the two features, which are required for computing concept definitions, i.e., ontology reasoning and proof tracing. In contrast, DeFind relies on external reasoning and proof tracing services. The question whether it suffices to compute a single concept definition strongly depends on application. For instance, the method from [3] is employed in [5] to compute query reformulations. In [6], checking for existence of a concept definition (or computing a single one) is used for ontology decomposition. In both applications, the quality of the obtained result is strongly related to the form of a computed definition. On the other hand, it is argued in [7] that having multiple definitions is helpful in the context of ontology alignment, since having a choice of several (semantically equivalent, but syntactically different) definitions facilitates finding matches between concepts from different ontologies. The authors propose a heuristic approach for computing concept definitions based on examination of concept definition patterns, which typically occur in ontologies. This solution does not employ ontology reasoning and is in general faster, but it can miss some definitions computed by the methods that employ proof tracing.

6 Outlook

The exponential lower bound for the size and number of shortest concept definitions is shown in this paper by an artificial ontology example. However it demonstrates the natural phenomenon: some concepts can be used as auxiliary ones in ontology to make definitions shorter. If one asks whether there is a definition, which does not contain certain concept or role names, then one may get a positive answer, but the obtained definition may be long and not easy to comprehend. It is to be understood whether a significant increase of definition size can happen in real-world ontologies. Further, it may be important to develop techniques that help to automatically suggest extensions of the target signature, when a blow-up is unavoidable. One of the primary goals of further implementation is to support other language features of OWL 2 EL profile of the Web Ontology Language, e.g., inclusion and composition of roles. These features are particularly interesting, because on one hand they are frequently used in ontologies and on the other hand, Theorem 1 is no longer true for $\mathcal{EL}$ extended with these features. Finally, there is a room for optimizations for DeFind. If the plugin is used extensively to compute definitions of the same concept wrt different signatures then it makes sense to implement optimizations proposed in [4], which employ incremental reasoning to reduce computation overhead, when only a small part of the input is changed.

References

1. Kazakov, Y., Krötzsch, M., Simancik, F.: The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. J. Autom. Reasoning **53**(1) (2014) 1–61
2. Kazakov, Y., Klinov, P., Stupnikov, A.: Towards reusable explanation services in protege. In Artale, A., Glimm, B., Kontchakov, R., eds.: Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017. Volume 1879 of CEUR Workshop Proceedings., CEUR-WS.org (2017)
3. ten Cate, B., Franconi, E., Seylan, I.: Beth definability in expressive description logics. J. Artif. Intell. Res. **48** (2013) 347–414
4. Ponomaryov, D.K., Vlasov, D.: Concept definability and interpolation in enriched models of $\mathcal{EL}$ -tboxes. In Eiter, T., Glimm, B., Kazakov, Y., Krötzsch, M., eds.: Informal Proceedings of the 26th International Workshop on Description Logics, Ulm, Germany, July 23 - 26, 2013. Volume 1014 of CEUR Workshop Proceedings., CEUR-WS.org (2013) 898–916
5. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation over databases with first-order and description logics ontologies. J. Artif. Intell. Res. **48** (2013) 885–922
6. Konev, B., Lutz, C., Ponomaryov, D.K., Wolter, F.: Decomposing description logic ontologies. In Lin, F., Sattler, U., Truszczynski, M., eds.: Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010, AAAI Press (2010)
7. Geleta, D., Payne, T.R., Tamma, V., Wolter, F.: Computing minimal definition signatures in description logic ontologies. Technical report, University of Liverpool (2016)