User-aware Explications of Ontology Consequences: Levelling Technicality

Rafael Peñaloza[1][0000−0002−2693−5790] and
Anni-Yasmin Turhan[2][0000−0001−6336−335X]

[1] University of Milano-Bicocca, Italy rafael.penaloza@unimib.it
[2] TU Dresden, Germany Anni-Yasmin.Turhan@tu-dresden.de

Abstract. Explaining consequences obtained from ontological reasoning is an active research topic. Unlike the computation of the cause of the consequence, the explication of the cause has received little attention so far. However, as many ontologies are designed by experts, the terms and notions used in an explanation need not be known to the user of the ontology-based system, before they can attempt to understand the underlying logical process.
In this paper, we address the task of making an explanation of a consequence more comprehensible to a variety of users by re-phrasing it in a vocabulary known to them. Assuming the existence of a dictionary, we attempt to rewrite technical jargon into expressions using a simpler vocabulary. We show that solving this problem requires taking several technical issues into account.

Keywords: Explications · Description Logics · user-awareness.

1 Introduction

With the deployment of artificial intelligence approaches in industrial applications, there is a growing need to explain the decisions made by artificial agents to external users who might not be fully knowledgeable about the system and its internal mechanisms. This need has given birth to the field of explainable AI (XAI) [8]. There are two main interpretations of XAI: explain *how* the AI system reached its conclusions—more akin to debugging—or explain *why* was the final conclusion reached. In this work, we consider the latter which, in the context of logic-based knowledge representation and reasoning (KR), usually refers to enumerating the logical constraints or the logical steps that yield a given consequence.

What constitutes a good explanation from the point of view of the social sciences was recently summarized in [11]. To build a useful explanation, one needs to solve two tasks. One is to *extract* the information that we want to provide—*what* does the explanation say? This is usually known as the *attribution* step [11]. The other is to identify *how* to express this information to make it intelligible. This second step of developing an *explication* of the information to be presented is at the heart of most XAI approaches in KR. During the attribution phase,

most methods identify a minimal portion of the knowledge base which yields the consequence (a so-called *justification*). The explication phase then transforms this justification into a more comprehensible version.

Some approaches generate logical proofs of different shapes, to allow the user to understand the derivation steps made [1]. A different approach is to rewrite the logical statements from the justification into natural language [2]. While quite promising, these approaches suffer from two main drawbacks: (i) they are user agnostic; that is, they generate the same explication independently of the requirements of the explainee; and (ii) they are limited to the vocabulary explicitly appearing in an ontology. The latter issue is specially important as finding the right words to use is a fundamental step of human explanations [7]. Indeed, the wrong terminology may hinder understanding of otherwise simple notions. For example, a user of an ontology-based biology application may be puzzled by the occurrence of notion "Zaglossus Bruijni"—which they do not understand—in the explication provided to them. In particular, if the more familiar term "Echidna" could be used instead. Importantly, we cannot expect knowledge bases or *ontologies* (which are constructed and used by domain experts) to avoid technical jargon and, at the same time, different users have differing expertise and explanatory needs.

We propose user-aware generation of explications for ontology consequences. This general task poses two challenges to be addressed. The first is to use terminology that is appropriate for the explainee in terms of being neither too complex nor overly simplistic. The second challenge is to choose an appropriate structure for the explication that does not confuse the explainee—neither by the use of complex logical expressions nor by its sheer length. We deem the first challenge more fundamental and address it in this paper. Our approach for user-aware generation of explications is based on a dictionary, which allows us to transform technical jargon appearing in a justification into logically-equivalent, but easier to understand expressions. Users can fine-tune their explication by specifying a collection of terms that they understand, representing their "level of technicality." Intuitively, the dictionary is stratified according to technicality levels for different users. Our approach is orthogonal to proof generation and translation to natural language. Indeed, an ideal explanation approach would combine all three of them, adding further user-aware flexibility.

As a first step to provide user-aware explications, we consider a setting where the user provides the vocabulary they are able to understand, and the explanation should be restricted to this vocabulary only. From an abstract point of view, this idea is very similar to the task of *forgetting* (or uniform interpolation) [9, 10], where the goal is to construct a knowledge base that "forgets" a class of symbols—in our case, the symbols that are not understood by the user. Uniform interpolation tries to preserve all the information of the knowledge base that refers to the wanted vocabulary, while completely removing any explicit reference to the vocabulary to be forgotten. In our case, though, we want to preserve the knowledge of technical terms, but express it through a simpler vocabulary. An important difference between these two problems, which will become clear

in Section 3, is that forgetting treats the whole vocabulary equally, while in our case we do not want to *oversimplify* the explication. Thus, from different potential explications, we prefer one that is maximally "technical" while remaining understandable to the user.

In this paper, we build the foundations needed to tackle this problem. Starting from a very strict first formalisation, based on concept equivalences, we show through a series of examples that any available method should take into account some technicalities that may not seem obvious at first sight. We thus show that solving the problem is not only useful for the area of XAI, but also interesting from a technical point of view in its own right.

2 Approaching the Problem of User-aware Explications

An important aspect of explaining a consequence to a user, which is not yet addressed sufficiently in the knowledge representation and reasoning field, is to take into account *who* the user is, and *how* to better approach them. There is certainly no universal solution to this, as different users have varying levels of understanding and explanatory needs. This motivates our interest in methods providing *user-aware explanations*; that is, methods to generate explanations targeting different user's needs.

We consider the case where knowledge is encoded in an ontology using a description logic (DL) [4]. In a nutshell, DLs use as building blocks two countable disjoint sets N_C and N_R of *concept names* and *role names*, respectively. Complex *concepts*, which correspond to unary predicates of first-order logic, are built with the help of different constructors. As a prototypical specimen we consider the light-weight DL $\mathcal{EL}$ [3] whose concepts are built according the grammar rule

$$C ::= \top \mid A \mid C \sqcap C \mid \exists r.C \ ,$$

with $A \in N_C, r \in N_R$, and its sublogic $\mathcal{L}_0$ which disallows the $\exists r.C$ constructor. Knowledge in DLs is expressed through an *ontology*: a finite set of *axioms*, of which the most common are so-called *GCIs* of the form $C \sqsubseteq D$ with C, D concepts. The semantics of this logic is defined in terms of *interpretations*, which are pairs $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function* that maps each concept name $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to complex concepts by setting $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} := \{\delta \in \Delta^{\mathcal{I}} \mid \exists \eta \in C^{\mathcal{I}}.(\delta, \eta) \in r^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ is a *model* of the ontology $\mathcal{O}$ iff for every GCI $C \sqsubseteq D \in \mathcal{O}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. The GCI $C \sqsubseteq D$ is a *consequence* of the ontology $\mathcal{O}$ iff every model of $\mathcal{O}$ is also a model of $\{C \sqsubseteq D\}$ and is denoted by $\mathcal{O} \models C \sqsubseteq D$. We often use the term "entailment" to refer to consequences as well. For full details on DLs see [4].

GCIs in an ontology act as *constraints* on the class of relevant interpretations, i.e. the interpretations. In particular, this means that consequences are *monotonic*; that is, if φ is a consequence of the ontology $\mathcal{O}$, then φ is necessarily

a consequence of any ontology containing all the GCIs in $\mathcal{O}$. In the following, we also use two special kinds of GCIs. A *concept equivalence* is of the form $C \doteq D$ where C, D are concepts. It abbreviates the two GCIs $C \sqsubseteq D, D \sqsubseteq C$ and hence is satisfied whenever $C^{\mathcal{I}} = D^{\mathcal{I}}$. A *concept definition* is a concept equivalence $A \doteq D$ where $A \in N_C$; that is, the left-hand side consists of a concept name only.

Given a consequence φ of the ontology $\mathcal{O}$, a user may want to know why this is a consequence. One early approach to answer this question was to provide a so-called *justification*; that is, a (subset) minimal sub-ontology of $\mathcal{O}$ from which φ still follows. The existence of one or more of such minimal subsets is guaranteed by the monotonicity of the logic. The set of axioms forming a justification can be seen as a crude explanation for the entailment of φ. It provides sufficient information about the derivation (the justification still entails the consequence) and avoids any superfluous information given by irrelevant GCIs. Importantly, a single consequence may have several (even exponentially many) justifications [6].

However, by their definition, justifications are always built with axioms from the original ontology. Developed and maintained by experts, and with different agendas in mind, the GCIs in an ontology tend to use a quite technical, or even private, terminology which might not be intelligible to the user of the ontology requiring an explanation of the consequence. Following the terminology used in the introduction, justifications realize the *attribution* (or information extraction) step of explanation development. Our goal, instead, is to provide an *explication*: a variant of such a justification which is understandable by the user. In slightly more formal terms, we are interested in the following problem.

Problem 1 (justification explication). Given a justification $\mathcal{J}$ and a user u, construct an equivalent ontology $\mathcal{E}$ using only terms understandable by u.

Of course, the exact specification of $\mathcal{E}$ depends on specifying what it means to be understandable to a user u, their available terminology, and the capability of rewriting technical concept and role names into more comprehensible ones. Depending on the specific scenario, an ontology $\mathcal{E}$ as required by Problem 1 might not exist. This issue can be slightly alleviated by recalling that a given consequence may have several justifications. Thus, we consider also a more general version of the problem.

Problem 2 (consequence explication). Given an ontology $\mathcal{O}$, a consequence φ, and a user u, construct an ontology $\mathcal{E}$, equivalent to some justification of φ on $\mathcal{O}$, using only terms understandable by u.

Importantly, Problem 2 may still have no solution, if none of the available justifications can be "explained" in terms understandable to the user. Hence, we are also interested in finding conditions where the problems are guaranteed to have a solution, or in efficiently identifying whether a solution exists.

A technically simple, but not necessarily optimal method for solving Problem 2 is to enumerate all possible justifications, and try to solve Problem 1 on them, until an adequate ontology $\mathcal{E}$ is found, or it is deduced that none exists.

Such an approach introduces a polynomial overhead on the number of justifications (which, as mentioned, may be exponential) in the case of $\mathcal{L}_0$. For $\mathcal{EL}$ ontologies, an exponential overhead is unavoidable even in cases with only polynomially many justifications [12]. In the following, we focus on Problem 1 only.

In the next section we provide a first full formalisation of this problem, starting from a formal definition of the notion of "understandability" that we consider here. We also show, through a series of examples, that even in the limited scenario considered, one must pay careful attention to avoid sub-optimal or plainly wrong solutions.

3 A First Solution

In order to solve the problems described in the previous section, we must first be able to express variations of technical notions at different levels of interpretability, along with the limits of understandability of each individual user. In this way, adequate terms to present to the user guaranteeing that they are understood can be selected automatically.

For this first solution, we classify users by means of their technical expertise, and in particular of the vocabulary that they can understand. Hence, for example, depending on their expertise, users may understand better or prefer to use the (equivalent) terms "monotremata," "oviparous mammals," or "egg-laying, milk-producing animals." We thus consider two sources of knowledge. First, we have a *domain ontology* $\mathcal{O}_D$, which encodes all the knowledge of the domain. This ontology, which is usual encountered in knowledge-aware applications, is maintained by experts and hence assumed to use as precise and technical a terminology as needed for an adequate modelling of the domain knowledge. Second, we have a *technical ontology* $\mathcal{T}$, which models relationships between more *technical* (or more complex) terms, and terms often used at lower levels of expertise. Before formally defining the technical ontology, we need to specify the technicality classes.

A *vocabulary* is a finite set $\mathsf{Voc} \subseteq N_C \cup N_R$ of concept and role names. A *technical vocabulary* is a partially ordered vocabulary $(\mathsf{Voc}, \preceq)$ where $\alpha \preceq \beta$ expresses that α is *at most as technical as* β. Thus, someone with greater expertise in the area would use terms that are larger w.r.t. $\preceq$. Importantly, the use of a partial order admits to have incomparable terms (where none is more technical than the other) and terms that are equally technical. Each element $\alpha \in \mathsf{Voc}$ defines a *technicality class*

$$\lceil \alpha \rceil := \{\beta \in \mathsf{Voc} \mid \beta \preceq \alpha\}$$

containing all the terms from Voc which are at most as technical as α. The idea is that a user capable of understanding α can also understand all terms in $\lceil \alpha \rceil$. Given a concept C, $\mathcal{V}(C)$ denotes the set of all concept and role names appearing in C; i.e., the vocabulary of C. This notion is extended in the obvious manner to GCIs and ontologies: $\mathcal{V}(\mathcal{O})$ is the set of all names appearing in the ontology $\mathcal{O}$. Since users need to understand all the terms appearing in a GCI, justification

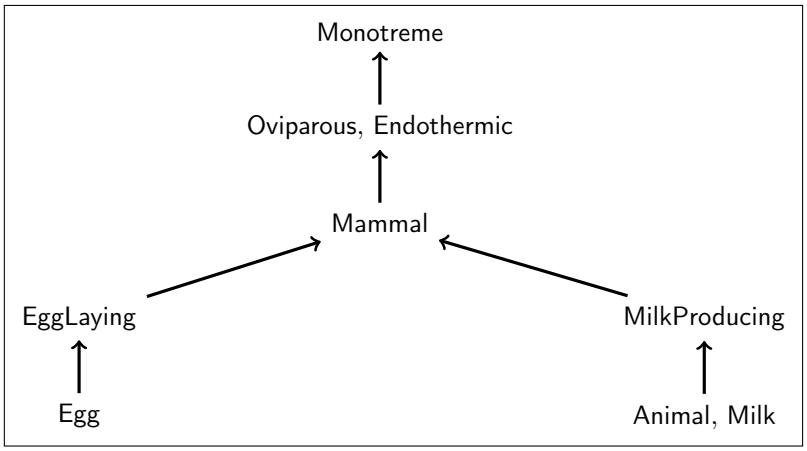

Fig. 1. A technical vocabulary including incomparable and equivalent terms.

or ontology, we associate their expertise (or *technicality level*) to technicality classes.

Example 3. Figure 1 depicts a Hasse diagram for a technical vocabulary. For instance, $\mathsf{Mammal} \preceq \mathsf{Oviparous}$ means that Mammal is not more technical than Oviparous. Since they do not appear together in the diagram, Mammal is in fact *less* technical. Animal and Milk are equally technical, and incomparable to Egg. The technicality class of Mammal is

$$\lceil \mathsf{Mammal} \rceil := \{\mathsf{Mammal},\ \mathsf{EggLaying},\ \mathsf{Egg},\ \mathsf{MilkProducing},\ \mathsf{Animal},\ \mathsf{Milk}\}.$$

After these preliminaries, we can now define the technical ontology.

Definition 4 (technical ontology). *Let* $(\mathsf{Voc}, \preceq)$ *be a technical vocabulary. A* technical ontology $\mathcal{T}$ *is a finite set of concept definitions using only terms in* Voc*, such that for every concept definition* $A \doteq C \in \mathcal{T}$ *it holds that* $\mathcal{V}(C) \subseteq \lceil A \rceil$.

A technical ontology provides definitions for some of the terms in the vocabulary, which are based on terms of lesser or equal technicality level. In simpler terms, $\mathcal{T}$ it is a stratified *dictionary* of concept names. The idea is that through those definitions, one can "rewrite" terms that a user may not understand into expressions using a "simpler" vocabulary. Ideally, this rewriting should contain expressions which use only concept and role names intelligible for the user.

Example 5. Using the technical vocabulary from Example 3, we can define the technical ontology

$$\begin{aligned}\mathcal{T} := \{\ \mathsf{Monotreme} &\doteq \mathsf{Oviparous} \sqcap \mathsf{Mammal},\\ \mathsf{Mammal} &\doteq \mathsf{Animal} \sqcap \mathsf{MilkProducing},\\ \mathsf{Oviparous} &\doteq \mathsf{Animal} \sqcap \mathsf{EggLaying} \qquad \},\end{aligned}$$

which expresses that monotremata are oviparous mammals; mammals are milk-producing animals; and oviparous are egg-laying animals. Note that the condition of decreasing technicality is satisfied by these definitions.

With the technical ontology in place, we can now formalise the general problems introduced in Section 2. To specify the level of technicality, we use the technicality classes defined by the technical ontology. That is, a user selects a term $\alpha \in \mathsf{Voc}$ such that they understand all terms in $\lceil \alpha \rceil$.

Problem 6. Let $\mathcal{J}$ be a justification, $(\mathsf{Voc}, \preceq)$ a technical vocabulary, $\mathcal{T}$ a technical ontology, and $\alpha \in \mathsf{Voc}$. Construct an ontology $\mathcal{E}$ such that

1. $\mathcal{E} \cup \mathcal{T}$ and $\mathcal{J}$ are equivalent, i.e. have the same models and
2. $\mathcal{V}(\mathcal{E}) \subseteq \lceil \alpha \rceil$.

If it exists, the ontology $\mathcal{E}$ is called an *α-bounded explication* of $\mathcal{J}$.

As defined, Problem 6 leaves quite some liberty on its solutions: the resulting ontology $\mathcal{E}$ may have nothing in common with the original justification $\mathcal{J}$, as long as it is logically equivalent, modulo $\mathcal{T}$. To make it more tractable, we note that it suffices to manipulate the concept names to an understandable form.

Proposition 7. *Let $\mathcal{J}$ be a justification, $(\mathsf{Voc}, \preceq)$ a technical vocabulary, $\mathcal{T}$ a technical ontology, and $\alpha \in \mathsf{Voc}$. There exists an α-bounded explication of $\mathcal{J}$ if*

- *every role name $r \in \mathcal{V}(\mathcal{J})$ is such that $r \preceq \alpha$, and*
- *for every concept name $A \in \mathcal{V}(\mathcal{J})$ there exists a concept C s.t. $\mathcal{T} \models A \doteq C$ and $\mathcal{V}(C) \subseteq \lceil \alpha \rceil$.*

The proof of this proposition is straightforward, and hence left out of this paper. The explication can be constructed by simply substituting every concept name in $\mathcal{J}$ by their equivalent C. Note that the converse is not necessarily true: there could exist α-bounded explications which are not constructed through substitutions of concept names. Taking advantage of this proposition, we now focus on the simpler problem of finding a concept explication.

Definition 8 (concept explication). *Let $A \in N_C$, $(\mathsf{Voc}, \preceq)$ a be technical vocabulary, $\mathcal{T}$ a technical ontology, and $\alpha \in \mathsf{Voc}$. A concept C such that $\mathcal{T} \models A \doteq C$ and $\mathcal{V}(C) \in \lceil \alpha \rceil$ is called an* α-bounded explication *of A.*

Returning to our running example, we see that

$$\mathsf{Animal} \sqcap \mathsf{MilkProducing} \sqcap \mathsf{EggLaying}$$

is indeed an Oviparous-bounded explication of Monotreme—in fact, it is even a Mammal-bounded explication. However, for a user who understands the term Oviparous, expanding out the definitions to the point of EggLaying may be counterproductive as it makes the explication verbose and not to mention the possibility of taking offence from an overly simplistic explanation. Thus, rather than presenting the user with an arbitrary α-bounded explication, we want to find

one which is as technical as possible, while remaining understandable. In our example, a more adequate explication (from this point of view) of Monotreme is Oviparous $\sqcap$ Mammal. In general, we will prefer *optimal* explications that are as technical as possible within the selected technicality class.

Definition 9 (optimal explication). *We extend the ordering $\preceq$ to complex concepts by setting $C \preceq D$ iff for every term $v \in \mathcal{V}(C)$ there is a $w \in \mathcal{V}(D)$ such that $v \preceq w$. C is* more technical than *D (denoted $C \prec D$) iff $C \preceq D$ but $D \not\preceq C$.*

An α-bounded explication D of C is optimal *if there exists no α-bounded explication D' of C which is more technical than D.*

A first idea to try to find (optimal) explications of concepts would be to follow a term rewriting approach [5]. Indeed, each concept definition $A \doteq C$ in the technical ontology can be seen as a rewriting rule which substitutes the concept name A with a complex expression using the symbols from C. These symbols may be further rewritten with other complex expressions, following additional definitions in technical ontology. Under this view, all symbols in $\lceil\alpha\rceil$ are *terminals*; that is, terms that will not be rewritten, since they are already understandable. All other symbols in Voc are non-terminals, and should be rewritten into simpler, potentially understandable terms.

This idea, while tempting, cannot work due mainly to three properties of the technical ontology and the rewriting:

1. concept definitions may be cyclic;
2. concepts may have multiple definitions; and
3. the equivalence in concept definitions works on both directions.

The first issue can be solved by requiring that technical ontology is acyclic or by implementing a cycle detection method. The other two issues are worth exploring further.

3.1 Non-determinism Induced by Multiple Definitions

Since one concept name may have more than one associated definition in the technical ontology, it is not clear *a-priori* which definition should be used in the rewriting of a concept. Indeed, a wrong choice may result in a concept that is not an α-bounded explication, but which cannot be further simplified, even in cases where adequate explications exist.

Example 10. Suppose that the technical ontology from Example 5 contains also the concept definition Monotreme $\doteq$ Oviparous $\sqcap$ Endothermic. If we want to find a Mammal-bounded explication of the concept name Monotreme, we can substitute the name Monotreme by either (i) the term Oviparous $\sqcap$ Mammal or (ii) the term Oviparous $\sqcap$ Endothermic. In the former case, further expanding the definition of Oviparous yields an understandable concept at the Mammal level of technicality. The latter case, however, yields the concept Oviparous $\sqcap$ Endothermic which contains a concept name (Endothermic) that does not belong to the technicality class $\lceil$Mammal$\rceil$, nor has an associated concept definition. Hence no adequate explication can be derived from it.

At the same time, it is unfeasible to preserve all possible expansions in the hope of deriving at least one α-bounded explication. A naive construction of all possibilities would generate exponentially many such concepts, thus requiring exponential time and space to find such an explication (or decide that it does not exist).

This issue can be at least partially solved with the help of a non-deterministic algorithm, which guesses, for every substitution step, which definition to use. This means that we can decide whether an α-bounded explication exists—and, indeed, construct it—in non-deterministic polynomial time. While this solves the issue of multiple-definitions, it does not guarantee that the constructed explication is optimal.

3.2 Sub-optimality by Limited Use of Concept equivalences

All concept definitions in the technical ontology are required to have, in the right-hand side, only terms that are at most as technical as the concept name in the left-hand side. However, beyond this requirement there is no restriction about the ordering of the terms in different definitions. Moreover, we are ultimately interested in constructing *equivalent* concepts, for which the semantics of concept definitions need to be carefully considered.

Recall that the semantics of $A \doteq C$ guarantee that A and C are equivalent. So far, we have used this as a unidirectional "rule" to substitute an occurrence of A by the concept C. However, logical equivalences also hold in the converse direction: substituting the concept C by the concept name A would be equally correct. At first, this may seem as an irrelevant insight as it implies *increasing*—rather than decreasing—the technicality level of the resulting concept; we are substituting a simple description with technical jargon. Still, optimal solutions might require such an interplay of technicality decrease and increase.

Example 11. Consider once again the technical vocabulary from Example 3, but now using the technical ontology

$$\begin{aligned}\mathcal{T} := \{\ &\mathsf{Monotreme} \doteq \mathsf{EggLaying} \sqcap \mathsf{MilkProducing} \sqcap \mathsf{Animal},\\ &\mathsf{Mammal} \doteq \mathsf{Animal} \sqcap \mathsf{MilkProducing},\\ &\mathsf{Oviparous} \doteq \mathsf{Animal} \sqcap \mathsf{EggLaying} \qquad \};\end{aligned}$$

that is, we have modified the first concept definition. Note that there is no non-determinism here, since every concept name is defined only once. If we want to produce a Mammal-bounded explication for the concept name $\mathsf{Monotreme}$, we expand its definition, obtaining the concept $\mathsf{EggLaying} \sqcap \mathsf{MilkProducing} \sqcap \mathsf{Animal}$. This is already an explication on the right level of technicality and can be understood by a user who understands Mammal. However, it is not optimal.

To find an optimal explication, observe that $\mathsf{MilkProducing} \sqcap \mathsf{Animal}$ is equivalent to Mammal in $\mathcal{T}$. Thus, we can simplify the previously constructed concept into $\mathsf{EggLaying} \sqcap \mathsf{Mammal}$, which is in fact an optimal explication for $\mathsf{Monotreme}$.

The issue is more complex. Even after finding a (potentially sub-optimal) α-bounded explication, it may still be impossible to find an optimal explication simply by increasing the technicality through a "converse" concept rewriting. As the following example shows, it may be necessary to further decrease the technicality of a concept, before it can be rewritten into an optimal explication.

Example 12. Consider again the technical vocabulary from Example 3, along with the technical ontology

$$\begin{aligned}\mathcal{T} := \{\ &\mathsf{Monotreme} \doteq \mathsf{EggLaying} \sqcap \mathsf{Mammal},\\ &\mathsf{Mammal} \doteq \mathsf{Animal} \sqcap \mathsf{MilkProducing},\\ &\mathsf{Oviparous} \doteq \mathsf{Animal} \sqcap \mathsf{EggLaying} \qquad \},\end{aligned}$$

and suppose that we want now an Oviparous-bounded explication of Monotreme. We first have no choice but to expand Monotreme to its unique definition. The resulting concept $\mathsf{EggLaying} \sqcap \mathsf{Mammal}$ is already an understandable explication to the desired level of technicality, but as we shall see, it is not optimal. In contrast to the previous example, it is not possible to substitute part of the explication with the concept Oviparous yet: although we have the conjunct EggLaying, we are still missing the term Animal in it. Notice that Mammal can be substituted by $\mathsf{Animal} \sqcap \mathsf{MilkProducing}$ yielding the (less technical) explication $\mathsf{EggLaying} \sqcap \mathsf{Animal} \sqcap \mathsf{MilkProducing}$, which can now be equivalently rewritten into the optimal explication (at the requested level of technicality) $\mathsf{Oviparous} \sqcap \mathsf{Mammal}$.

These simple examples highlight a very general limitation of the rewriting approach to finding explications. In order to guarantee optimality of the resulting concept, it may be necessary to cycle through several rounds of increased and decreased technicality. It is also possible to encounter cases where the technicality of a concept needs to be increased *beyond* the user-specified technicality level before it is later rewritten into an adequate explication; we leave the construction of such an example to the interested reader.

Example 12 implicitly showcases another issue that we have so far ignored, which refers to the idempotency of the conjunction. Note that in the last step of Example 12, we used the fact that Animal and $\mathsf{Animal} \sqcap \mathsf{Animal}$ are equivalent, in order to construct the two terms Oviparous and Mammal; each of them requires a mention of Animal in their definition. This means that some terms may need to be "recalled" after being used in a rewriting step to allow for another rewriting. The following example makes the issue explicit.

Example 13. Using the technical vocabulary from Figure 1, suppose that we are interested in explaining the concept name Monotreme to the technicality degree of Oviparous based on the technical ontology

$$\begin{aligned}\mathcal{T} := \{\ &\mathsf{Monotreme} \doteq \mathsf{EggLaying} \sqcap \mathsf{Mammal},\\ &\mathsf{Monotreme} \doteq \mathsf{Animal}\\ &\mathsf{Oviparous} \doteq \mathsf{EggLaying} \sqcap \mathsf{Animal} \qquad \}.\end{aligned}$$

If we try to rewrite terms using these definitions, we can transform Monotreme either to EggLaying ⊓ Mammal or to Animal. At this point, no other rewriting step is possible. However, since Monotreme is equivalent to EggLaying ⊓ Mammal, the latter concept is also equivalent to Monotreme ⊓ EggLaying ⊓ Mammal which can now be rewritten as Animal ⊓ EggLaying ⊓ Mammal, yielding the optimal explication Oviparous ⊓ Mammal.

4 Conclusions

Our goal is to explain to non-expert users the reasons why a consequence follows from a domain ontology. In contrast to other approaches where the emphasis is in the presentation of the axioms and explanation of the logical steps followed, we are more interested in explaining existing justifications using terms that are understandable to the user. To this end, we assume the existence of a dictionary, which defines technical jargon through equivalent, less technical expressions. Even at the restricted setting of this work, we have seen that finding optimal explications requires specialised techniques, which still need to be developed.

All our examples are based on the very simple logic $\mathcal{L}_0$. Clearly, additional issues arise if we extend the language to $\mathcal{EL}$ or beyond. For instance, it should be clear that *deciding* whether a given concept C is an optimal α-bounded explication of A is in co-NP for $\mathcal{L}_0$: if it is not, simply guess another concept D (which is a conjunction of concept names in $\lceil\alpha\rceil$) and verify that it is equivalent to A, and that $C \prec D$. This method does not work in $\mathcal{EL}$ because, due to the potentially nested existential restrictions, we have no guarantee that optimal explications are even of polynomial length.

As future work, we intend first to fully understand the properties of these problems in $\mathcal{L}_0$. Beyond finding tight complexity bounds for the decision problem and deriving effective algorithms for constructing one or all optimal explications, we will consider a more precise notion for a concept to be "more technical" than another. Another problem to consider is related to user education: if a term cannot be explained to the user's level of technicality, we want to propose a vocabulary—as close as possible to the user's own—which will allow them to understand it. Afterwards, we will look at $\mathcal{EL}$ and beyond to expressive DLs and other KR formalisms and entailments.

Another road of generalisation which is worth exploring is to relax the notion of explication and give up the requirement that it is equivalent to the original concept, but that it is similar enough. For instance, some of the conjuncts in the explication could approximate the original ones from above while others could be approximated from below. Finding out how to deal with such similarity will require to explore several alternatives.

References

1. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Finding good proofs for description logic entailments using recursive quality measures. In:

Platzer, A., Sutcliffe, G. (eds.) Proceedings of the 28th International Conference on Automated Deduction (CADE-28). Lecture Notes in Computer Science, vol. 12699, pp. 291–308 (2021). https://doi.org/10.1007/978-3-030-79876-5_17
2. Androutsopoulos, I., Lampouras, G., Galanis, D.: Generating natural language descriptions from OWL ontologies: the naturalowl system. J. Artif. Intell. Res. **48**, 671–715 (2013). https://doi.org/10.1613/jair.4017, https://doi.org/10.1613/jair.4017
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of IJCAI'05. pp. 364–369. Professional Book Center (2005)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
5. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
6. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic $\mathcal{EL}^+$. In: Proc. of KI'07. LNCS, vol. 4667, pp. 52–67. Springer (2007). https://doi.org/10.1007/978-3-540-74565-5_7
7. Bolognesi, M.: Where Words Get their Meaning: Cognitive processing and distributional modelling of word meaning in first and second language. John Benjamins (2020), https://www.jbe-platform.com/content/books/9789027260420
8. Doran, D., Schulz, S., Besold, T.R.: What does explainable AI really mean? A new conceptualization of perspectives. CoRR **abs/1710.00794** (2017), http://arxiv.org/abs/1710.00794
9. Eiter, T., Kern-Isberner, G.: A brief survey on forgetting from a knowledge representation and reasoning perspective. Künstliche Intell. **33**(1), 9–33 (2019). https://doi.org/10.1007/s13218-018-0564-6
10. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 989–995. IJCAI/AAAI (2011). https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-170
11. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1–38 (2019). https://doi.org/10.1016/j.artint.2018.07.007
12. Peñaloza, R., Sertkaya, B.: Understanding the complexity of axiom pinpointing in lightweight description logics. Artificial Intelligence **250**, 80–104 (2017). https://doi.org/10.1016/j.artint.2017.06.002