Improved Explanations in the Protégé OWL Ontology Editor

Cilliers Pretorius
prtpie003@myuct.ac.za
University of Cape Town
Rondebosch, South Africa

ABSTRACT

Description logics and formal ontologies represent knowledge in a way that allows for computational reasoning to generate conclusions. To increase and ensure user acceptance of these conclusions, explanations for why the conclusions were reached are required. Protégé is a popular ontology editor that uses the Web Ontology Language (OWL) developed for the Semantic Web. Protégé has the Explanation Workbench software tool, which provides justifications for inferred entailments in OWL ontologies. These justifications are difficult to understand for users not familiar with description logics and OWL. This paper details the design and implementation of improvements to the explanations provided by the Explanation Workbench. The theoretical framework and motivation for explanations are provided. A framework allowing ontology creators to define an explanation for an axiom is created and defined according to the OWL standards. The keywords of the Manchester Syntax used by Protégé are expanded and replaced with more natural language. This paper paves the way for future research into OWL and explanation facilities for it.

CCS CONCEPTS

• **Information systems → Web Ontology Language (OWL)**; Expert systems; • **Computing methodologies →** *Description logics*;

KEYWORDS

Explanation, OWL, Protégé, Manchester Syntax

1 INTRODUCTION

A logic-based reasoning system is a software system that generates conclusions that are consistent with a knowledge base (KB) or ontology. This ontology consists of a set of propositions that are defined to be true. There is an increasing trend to use logic-based reasoning systems as primary advisers in industries such as medicine, e-commerce, and finance to perform medical diagnosis, product recommendations, and stock market decisions. However, because the steps taken to generate a conclusion are usually hidden from the user, it cannot be guaranteed that the user accepts and acts upon the conclusion. [16]

This leads to systems that provide explanations and justifications as a key part of the system's design. [25] Research has shown that both novice and expert users greatly benefit from explanations, and designers should keep the intended user audience in mind when designing the system. [5]

Protégé is an ontology development tool that allows users to create ontologies according to the Web Ontology Language (OWL). OWL is a Description Logic (DL) that allows for precise and unambiguous definitions, allowing for reasoners to infer conclusions. [13] Protégé is one of the most popular ontology development tools, with users from a wide variety of backgrounds. [18]

It is bundled with the Explanation Workbench plugin developed by Horridge et. al. [13] The Explanation Workbench allows users to generate explanations for an inferred conclusion using the same reasoner. In particular, it generates justifications, which is defined as the minimal subset of the ontology sufficient for the entailment to hold. The output of this is simply the axioms that represent the ontology.

However, these axioms are difficult to understand if the user did not create the ontology themselves since it uses the Manchester Syntax of keywords in place of the mathematical symbols that are used in description logics. [13] Despite the use of keywords, both expert and novice users will struggle to understand the explanations unless they have enough knowledge of description logics. This is unlikely given that many ontologies are created for specific knowledge bases not related to description logics. [18]

This project attempts to provide more readable and more convincing explanations, built on the Explanation Workbench's explanations. In particular, it allows for users to see expanded keywords for the axioms[1], as well as allowing ontology creators to provide annotations that contain an explanation that will be displayed with the axiom.

This paper is the final report on the project. It states the aim of the project and provides the background to the project. Related work and the theoretical basis of description logics and explanation are described. The initial analysis and technical approach are described, followed by the implementation, evaluation, and discussion of the final product. The paper ends with conclusions and an exploratory overview of possible future research arising from this paper.

2 PROJECT AIM AND EXPECTED OUTCOMES

This project attempts to extend the Explanation Workbench Protégé plugin created by Horridge et al. [13] to produce explanations that are more readable and easily understood by users. The current tool attempts to generate readable explanations, but this occurs as a side-benefit of the syntax used by Protégé. It does not help users who are not familiar with description logics or ontologies in general.

With the overarching goal of improved readability and more effective explanations, two methods are considered. One method

[1] In this paper, axiom refers to a proposition in an ontology.

is to allow the creator of an ontology to define an explanation for an axiom, which would be displayed as the explanation for that particular axiom. The second method is to expand the keywords that are used in the axiom to use more natural language.

It is expected that the project will produce an extension to the Explanation Workbench that provides a framework for the ontology creator to define an explanation for an axiom. In addition, the keywords used by Protégé will be expanded to use more natural language. Thus, the explanations should be more readable and easily understood by users, even if they are unfamiliar with description logics.

3 BACKGROUND AND RELATED WORK

This paper and the screenshots provided will generally use Horridge's [11] example ontology regarding pizzas (the food item) for a more intuitive description of the theoretical basis.

3.1 Description Logics

Description logics are a family of knowledge representation languages. A description logic consists of a set of concept names and role names. A set of propositions (a knowledge base) are defined using atomic concepts and atomic roles, which are conveniently defined as unary predicates and binary predicates respectively. Propositions can be combined to form compound propositions, using logical structures such as negation, conjunction, and disjunction. [2]

The knowledge base is usually divided into the Tbox and the Abox. The Tbox is the set of axioms that describe the limits and structure of the knowledge base, while the Abox is the set of axioms describing a specific situation or interpretation in the framework defined by the Tbox. ([2], [27]) In the example description logic, the Tbox is the set of axioms describing how a pizza is defined, while the Abox is the set of axioms defining specific toppings and specific types of pizza.

With this formal structure based on predicate logic, algorithms that determine if a particular statement is true in a specific description logic can be created. [27] This can be extended to allow a reasoner to infer all axioms that are implied but not defined in the knowledge base.

3.2 Explanations

3.2.1 Motivation for Explanations. Explanations are considered a key component of most knowledge-based systems used by professionals. [14] The greatest benefit of explanations is usually the increased user acceptance of the system's recommendations, although this is not always the case. According to [21], including explanations in a system leads to a considerable increase in user acceptance of the system and its recommendations.

When novice users use explanations, they solve problems faster and more accurately than when not using explanations. [7] When expert users have explanations available, they accept the system's recommendations more readily and more fully. [1]

Explanations lead to greater trust in the system, and with this greater trust comes an increased likelihood that the system will be used repeatedly and that its recommendations will be followed. [23] Improved belief leads to solving problems using the system more accurately and in a shorter time. ([1], [8], [28])

3.2.2 Defining Explanations. Explanations vary depending on the system and what logical basis the system uses. Clancey [4] provided a detailed epistemology of explanations for rule-based systems, which defined three types of knowledge and types of explanation, generally corresponding to the types of questions a user might ask when using such a system. MYCIN, one of the first expert systems to feature explanations, provided multiple types of explanations. [6]

Since this project is focused on Protégé and its Explanation Workbench, only *Justification* explanations will be explored in detail here. Justifications are the most commonly used type of explanations in explanation tools. [1] Justification explanations provide the argument behind a reasoning system's conclusions. [24] In particular, a justification is defined as the smallest subset of the knowledge base where an entailment or conclusion is still valid. [12]

Justifications are used because of the simplicity of calculating them, as well as the clear link they have with the knowledge base. [13] The next section will clarify why the link is important to the explanation. Justifications can be calculated in two ways. Firstly, it can be done as an additional benefit to standard reasoning calculations as performed by any reasoner. Alternatively, it can be done separately with dedicated algorithms independent of any reasoner. This is commonly known as glass-box and black-box algorithms respectively. [9]

Justifications can be further refined to laconic and precise justifications. Laconic justifications are justifications where no axiom in the justification contains any superfluous parts, i.e., parts of compound axioms that are not required for the entailment to hold. Precise justifications are justifications that highlight parts of axioms that can be changed to ensure the ontology's consistency. [9] For this paper, only laconic justifications are of tangential interest.

3.2.3 Evaluating Explanations. In a project such as this, it is necessary to define a method of evaluating an explanation and its acceptability. Tintarev and Masthoff [26] provide a general structure of evaluating explanations with regards to the human aspect, with many of their criteria closely linked to the human-computer interaction fundamentals described by Nielsen.

However, Swartout and Moore [24] provide a specific method of evaluating explanations more relevant to description logics and Protégé. Fidelity, Understandability, Sufficiency, Low Construction Overhead, and Efficiency are the elements of a good explanation.

Fidelity focuses on how well the explanation represents what the reasoning system does. A better explanation has greater fidelity.

Justifications, with a clear link to the knowledge base, has high fidelity.

Understandability pinpoints whether the user understands the explanation that the reasoning system generates. This aspect is multifaceted because it relies on both the users' personal experience and the ability of the reasoning system to generate explanations at different levels of abstraction. The more understandable an explanation is, the better the explanation is.

Sufficiency focuses on whether the KB of the system has the required information that is needed to generate an explanation.

Low Construction Overhead. This element evaluates the difficulty of generating explanations. [24] suggest that it should not be more difficult to construct an explanation than it is to construct a reasoning system.

Efficiency concentrates on the impact that generating an explanation has on the runtime of reasoning systems. A good explanation consumes less runtime and is, therefore, more efficient.

This project is specifically focused on the Understandability aspect, also known as effectiveness in Tintarev's [26] criteria. This means a good explanation will require fewer additional explanations to convince the user.

3.3 The Web Ontology Language (OWL)

The Web Ontology Language is a language intended for information contained in documents to be processed by applications instead of humans. [17] It is an extension of the Resource Description Framework and is more expressive while still retaining computational completeness and decidability.

OWL has three variants:

- OWL DL corresponds to a true description logic, being the most expressive possible while guaranteeing computational completeness.
- OWL Lite is less complex, but simpler to interact with from a computational side.
- Lastly, OWL Full allows for the full expressiveness that is allowed by the Resource Description Framework, but computational completeness cannot be guaranteed. [17]

Something of particular note to this project is the fact that everything in OWL, whether classes, properties, or axioms, can have annotations. Annotations can themselves have annotations in an infinite spiral. These annotations can be of various types but are most often used as comments or labels. OWL DL limits annotations to a particular format, but any Annotation Property (i.e., type of annotation) can be defined as long as the format is followed.

Protégé uses OWL DL with a variant of the Manchester Syntax, which is a syntax developed to satisfy demands for "a less logician-like" syntax. [10] Hence, the terminology and axioms in this paper are likely to be less formal.

There have been several papers and considerable research establishing a theoretical base in explanations for OWL ontologies. Much of the work has focused on justifications. ([3], [15]) There has been some effort to generate explanations in forms closer to natural language. [19]

Horridge et al. [13] provided the OWL Explanation Workbench software library to calculate justifications for OWL ontologies, as well as a plugin for Protégé that implements the Explanation Workbench. Richard Power provided OWL Simplified English (OSE), which is an attempt at translating OWL statements and entailments to a natural language form. [20] The SWAT ontology editor[2] implements OSE and offers promise for future improvements to human interaction with OWL ontologies.

3.4 Protégé-OWL

The Protégé Ontology Editor was developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. It is a very popular ontology editor, with a large community still actively creating and contributing to the project. [18]

Given its role as one of the oldest and most popular ontologies, Protégé is often chosen as the testbed for new research in the fields of logic and AI. [18] The OWL Explanation Workbench, which is a software library for generating justifications in OWL ontologies, was first implemented as a plugin for Protégé. [13]

However, there has been no research regarding improvements to the explanations provided by Horridge's [13] tool. This might be due to the audience that Protégé seemingly enjoys, composed mainly of academics and domain experts. In this case, the users are generally informed about description logics and OWL. With the recent proliferation of commercial ontology editors, there appears to be a need for better explanations for Protégé specifically.

4 ANALYSIS AND DESIGN CHOICES

4.1 Protégé

The project's aims required familiarity with Protégé's functionality and how the implementation maps to the formal description of description logics.

4.1.1 Classes. The Protégé OWL Ontology Editor consists of the main application and several plugins that add additional functionality. To represent concept names in an ontology or description logic, Protégé uses classes. Classes can be nested to create subclasses, and all classes are subclasses of $owl : thing$. This subsumption of classes is transitive, i.e., $A\ SubClassOf\ B$ and $B\ SubClassOf\ C$ implies $A\ SubClassOf\ C$. Individuals are instances of classes. From the theoretical description logic point of view, individuals would be part of the Abox compared to the classes being part of the Tbox.

[2] http://mcs.open.ac.uk/nlg/SWAT/editor.html

4.1.2 Properties. To represent role names, Protégé uses Properties. This is separated into Object Properties and Data Properties. As with classes, each property is a subproperty of a root property, $owl : topObjectProperty$ and $owl : topDataProperty$ respectively in this case. Property subsumption is transitive. Properties can be limited to specific domains and ranges.

Object Properties are binary relations between two classes or class instances. In description logic, this would be $A.r\ B$, where class A is related through role r to class B. Object properties can have *functionality, inverse functionality, transitivity, symmetry, asymmetry, reflexivity,* and *irreflexivity* as characteristics. An example from the Pizza tutorial would be $Pizza\ hasTopping\ PizzaTopping$.

Data Properties are binary relations between a class and one of the OWL, RDF, or XML Schema Definition (XSD) data types, such as string, character, or integer. In description logic terms, these data types can be seen as concepts that are implicitly part of the ontology and the relation is between a class and an instance or two class instances. Unlike object properties, data properties can only be functional as characteristic. An example from the tutorial is $ExampleAmericanaPizza\ hasCalorificContentValue\ 359$.

4.1.3 Annotations. Several predefined annotation properties that can be added to anything, as per the OWL standards. Users can also define their own annotation properties, with more flexibility than is allowed by OWL in the strictest sense. The annotation values can be any of the OWL, RDF, or XSD datatypes.

Protégé uses the logical unary and binary relations that allow for compound concepts in description logic, namely negation, conjunction, disjunction, value restrictions, and existential restrictions. In the Manchester Syntax that Protégé uses, these are represented by the keywords *not, and, or, only*, and *some* respectively.

Protégé has the HermiT[3] reasoner bundled with it, and allows for other reasoners such as FaCT++ [27] to be added as plugins. When the selected reasoner is activated, the ontology is classified and evaluated for logical consistency. If the ontology is consistent, Protégé will display the inferred axioms as calculated by the reasoner.

4.2 Explanation Workbench

The OWL Explanation Workbench is a software library that allows justifications in OWL ontologies to be computed and displayed. It operates with both black-box and glass-box algorithms, using interfaces built into the library that allow suitably designed reasoners to be "plugged in".[13] The author of the library also created the Protégé plugin that implements explanation facilities (as calculated by the library) in Protégé.

The plugin creates a button for each axiom, labelled "Explain inference", that calculates all justifications for the axiom or entailment. The user can stop the calculation at will, even if all justifications have not been calculated. Once the calculation has stopped, the axioms contained in the justification will be displayed in a list. This list is a sublist of the listing that displays every justification for a particular entailment.

The user can choose whether to display regular or laconic justifications. By default, the regular justifications are selected. If the selection is changed, the justifications are recalculated according to the selection, unless the justifications are cached. The user can also limit the number of justifications that are displayed.

4.3 Manchester Syntax and Keyword Expansion

The Manchester OWL Syntax was developed to replace the variant of the German DL Syntax that was used by the most popular ontology editors at the time. [10] This syntax, while a vast improvement over the previous, still presents challenges to novice users and users not familiar with ontologies and description logics.

Hence, this project attempts to further improve the understandability of the Manchester Syntax by expanding the keywords to a more natural form. Table 1 lists some of the keywords of the syntax, along with the OWL DL concept it represents and the new expanded keywords this project replaces it with. For the sake of brevity, the full list of keywords and expansions can be found in the appendix. This list was compiled from the World Wide Web Consortium's (W3C) OWL Manchester Syntax documentation[4].

4.4 Explanatory Annotations

To generate more effective explanations, this project also looked at allowing ontology creators to specify an explanation for a particular axiom. Annotations in OWL can be of any format or datatype, hence it was chosen as the best way to allow creator defined explanations.

To allow for this, we define an annotation property exp:Explanation. In terms of W3's OWL standards, exp:Explanation is defined by the typing triple

```
<owl:AnnotationProperty rdf:about="exp:Explanation"/>
```

However, this definition is used only because the OWL standards do not contain a formal definition for explanations. If a formal definition or AnnotationProperty is added to the OWL standards, then the definition used in this project should be changed to reflect the OWL standards.

In the example ontology, an explanatory annotation for the axiom *AmericanaPizza SubClassOf hasTopping some MozzarellaTopping* could be *exp:Explanation : Americana pizzas have the mozzarella topping*. This allows for a much more natural explanation for the axiom than just the axiom and justification alone.

[3] See [22] for the initial paper describing the reasoner.

[4] https://www.w3.org/TR/owl2-manchester-syntax/

OWL DL Concept	Manchester Syntax	Expanded Keyword
$A \subseteq B$	A SubClassOf B	A is a subclass of B
$A \subseteq B$	A Type B	A is a type of B
$A \equiv B$	A EquivalentTo B	A is equivalent to B
$A = B$	A SameAs B	A is the same as B
$r \geq i\ B$	r min i B	r no less than i B
$r \leq i\ B$	r max i B	r no more than i B
$r\ X$	r value: X	r with the value X
$r \subseteq s$	r SubPropertyOf s	r is a subproperty of s
r Domain A	r Domain A	r has the domain A
r Range A	r Range A	r has the range A
$\neg A$	not A	not A
$A \sqcap B$	A and B	A and B
$A \sqcup B$	A or B	A or B
$\exists .r\ B$	r some B	r at least one B
$\forall .r\ B$	r only B	r only B
$q = r^{-1}$	q InverseOf r	q is the inverse of r
Functionality	Functional: r	The property r is functional
Transitivity	Transitive: r	The property r is transitive
$A \cap B = \varnothing$	DisjointClasses: A, B	The classes A and B are disjoint classes.
$A = \bigcup_{i \in I} A_i$	A DisjointUnionOf A1, A2	A is the disjoint union of A1 and A2.

Table 1: A table listing some of the keywords from the Manchester Syntax and the expanded keywords used in this project.

5 IMPLEMENTATION

The full description of how the Explanation Workbench plugin was implemented can be found in [13]. This project changes nothing regarding the original plugin's functionality, and only adds to it. Explanations are still invoked when the user clicks on an "Explain inference" button that appears next to each axiom.

Checkboxes were added to allow the user to decide if annotated explanations should be displayed or keywords expanded. Users are allowed to have the checkboxes checked in any combination. If neither is checked, the output is exactly as the original Explanation Workbench would produce. Figure 1 shows the output when no checkbox is selected to explain the example inference of *AmericanaPizza SubClassOf CheesyPizza*.

5.1 Expanded Keywords

Keyword expansion is implemented as a single function. If the checkbox for expanded keywords is checked, this function receives an axiom from the renderer that displays the axiom on the explanation panel. The function splits the axiom (a String object) according to whitespace. It then iterates through the tokens created and if it finds a keyword listed in Table 1, it replaces the keyword with the expanded language. It returns the expanded axiom to the renderer to be displayed on the screen.

Using the same example inference as Figure 1, Figure 2 shows the output when only expanded keywords are selected. Unfortunately, the keyword expansion means that the Protégé renderer does not highlight the syntax with colour. Since the current renderer utilises a tokenizer, it will be near impossible to have the current renderer add colour to the expanded keywords. This would have required a volume of work beyond the scope of this project.

5.2 Explanatory Annotations

The explanatory annotations are displayed through a small change to the original render method. If the checkbox for explanatory annotations is checked, the method checks if the axiom has any annotations attached to it. If there is at least one annotation, it iterates over all annotations and checks if the annotation has the annotation property corresponding to "exp:Explanation".

If the axiom has an explanatory annotation, the renderer will display the axiom (in its original Manchester Syntax form) and append the explanatory annotation after it. This is done to provide context to the annotation. Figure 3 shows the same justifications as Figure 1 with the explanatory annotations checkbox checked.

Note that this functionality requires the ontology creator to have defined the explanatory annotations beforehand. All axioms do not require an annotation, it might well be sufficient to have only the less intuitive axioms annotated. Also, if the user selects laconic justifications, the Explanation Workbench generates new axioms that are not linked to the annotations of the original. Thus, laconic justifications will only have the keywords expanded and will not display any explanatory annotations.

5.3 Action Priority

This project assumes that the explanations provided by the ontology creator will always be superior to the axiom with expanded keywords. Therefore, if both checkboxes are selected as in Figure 4, the renderer will first look if the axiom has any suitable annotations.

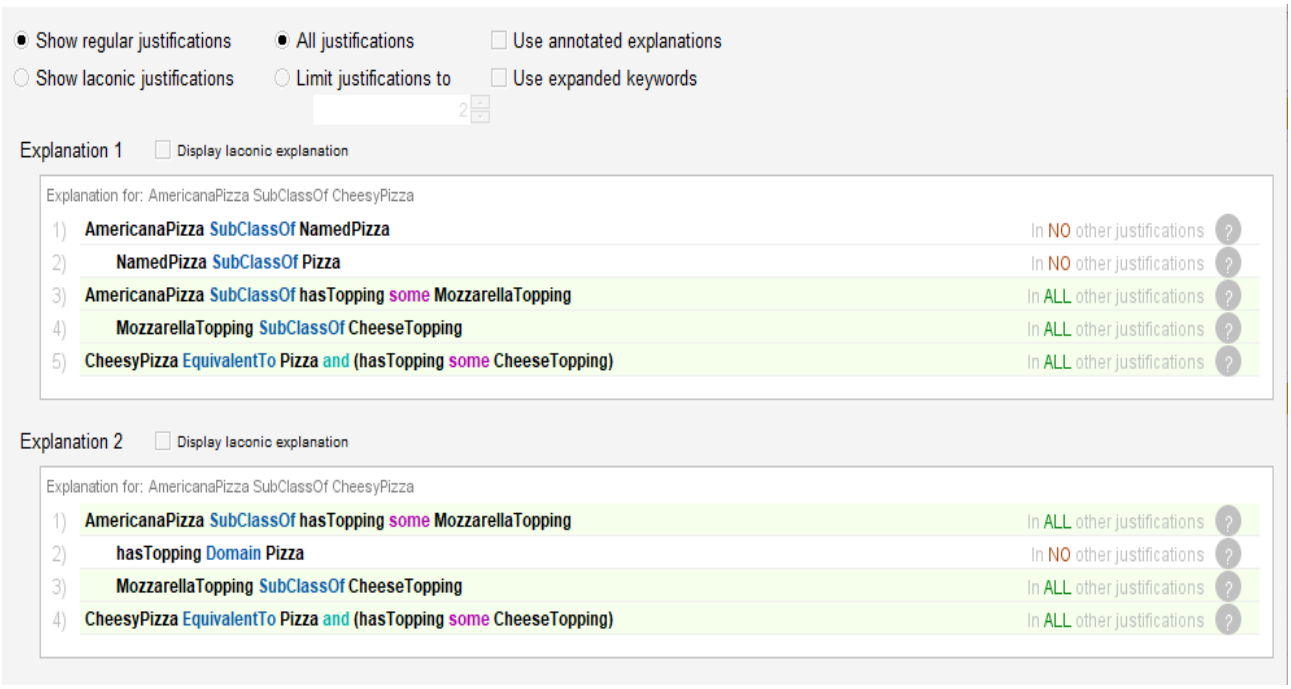

Figure 1: Output of the Explanation Workbench for an example inference. Note the checkboxes are unchecked and therefore the output is as the original facility would provide.

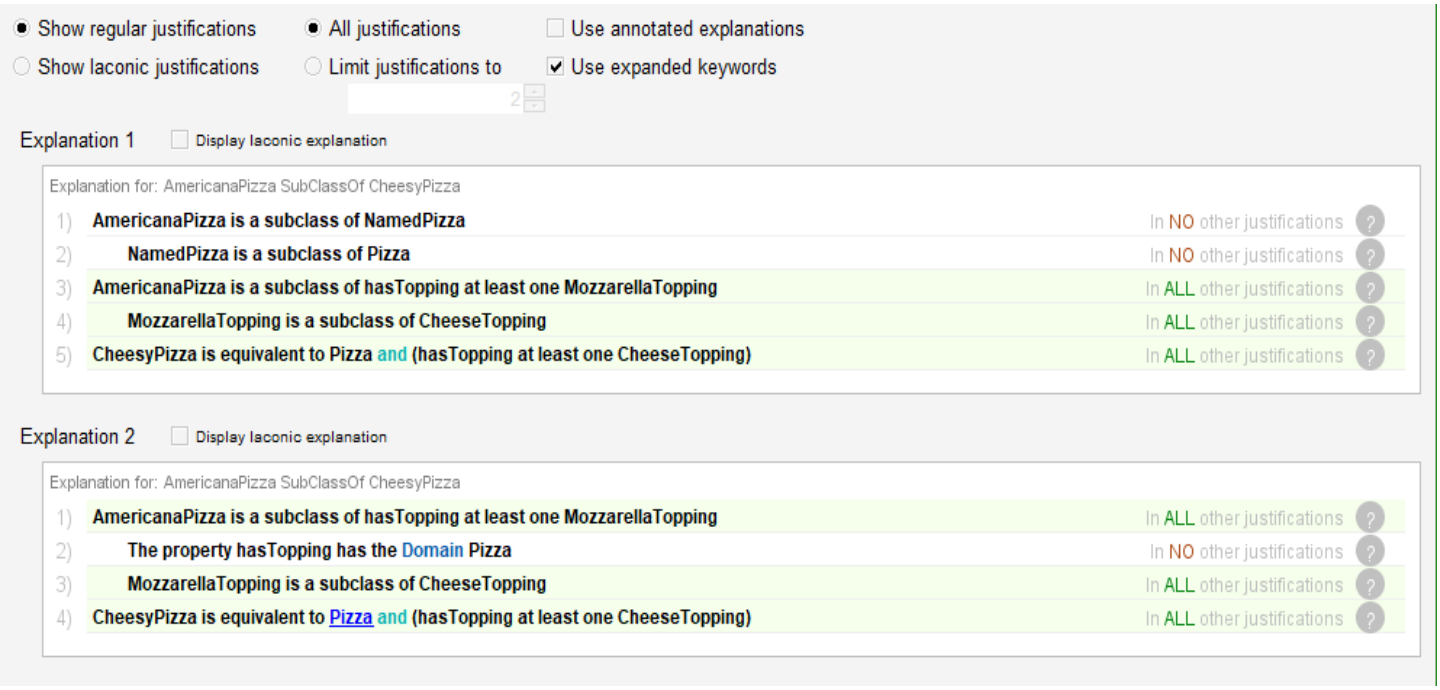

Figure 2: Output of the Explanation Workbench for an example inference, with the keywords expanded as per Table 1's specifications.

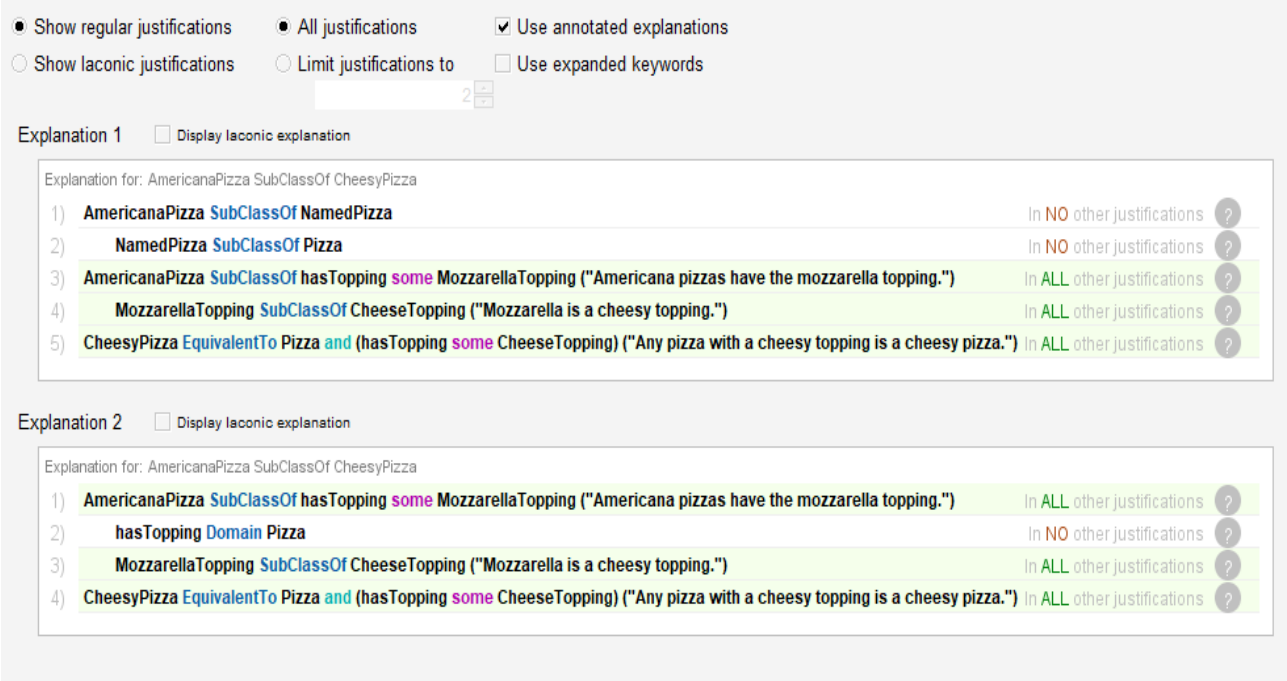

Figure 3: Output of the Explanation Workbench for an example inference, with explanatory annotations appended to the axioms that have suitable annotations.

Only if the axiom has no suitable annotations will the renderer expand the keywords of the axiom.

6 RESULTS AND EVALUATION

The project successfully produced a framework whereby an ontology creator can define an explanation for a particular axiom using the explanatory annotations. Figure 3 shows the explanatory annotations in use, and it clearly demonstrates increased readability in comparison to the original Explanation Workbench.

The keyword expansion was successfully implemented, as can be seen in Figure 2. The expansions are patterned on the suggested reading of the description logic symbols suggested by the Baader et al. [2] textbook on description logics. This assists not only in readability, but also increases understanding of description logics and therefore the users' comprehension of the explanations. The method that expands the keywords is separate from the Explanation Workbench and can, therefore, be accessed in any project that wants to expand the OWL Manchester Syntax keywords.

By allowing the user the option to choose whether the keywords are expanded or the explanatory annotations are displayed, this extension caters to all users. It allows users who are accustomed to the current Explanation Workbench output to continue using their preferred output while allowing novice users to have a less formal and less complex notation for their output. This increases the effectiveness of the explanation since they are more likely to understand it.

Since there is no calculation involved, only string manipulation, the program's runtime isn't affected in any noticeable manner. It allows ontology creators to define a more thorough and encompassing explanation for axioms, which will increase the effectiveness of the explanation for both expert and novice users. This project achieved the objectives set out at the start and is thus a success.

6.1 Limitations

Despite the project's overall success, there is still potential for improvement. Currently, the explanatory annotations need to be manually added by the ontology creator. However, many ontologies are not in active development anymore and it will be a substantial undertaking to add annotations to all axioms in all ontologies. In addition, many OWL ontologies are created automatically, without human interaction. These generated ontologies will not have explanatory annotations included.

The keyword expansion was done without any testing and evaluation by users, and there might be further enhancement possible in the natural language used for this expansion.

7 USAGE

The source code of this extension of the Explanation Workbench can be found and freely downloaded from the project's GitHub repository[5]. It will require Protégé to be of use, which can be downloaded from `protege.stanford.edu`.

[5] `https://github.com/Pietersielie/Explanation-Workbench-More-Readable-Extension`

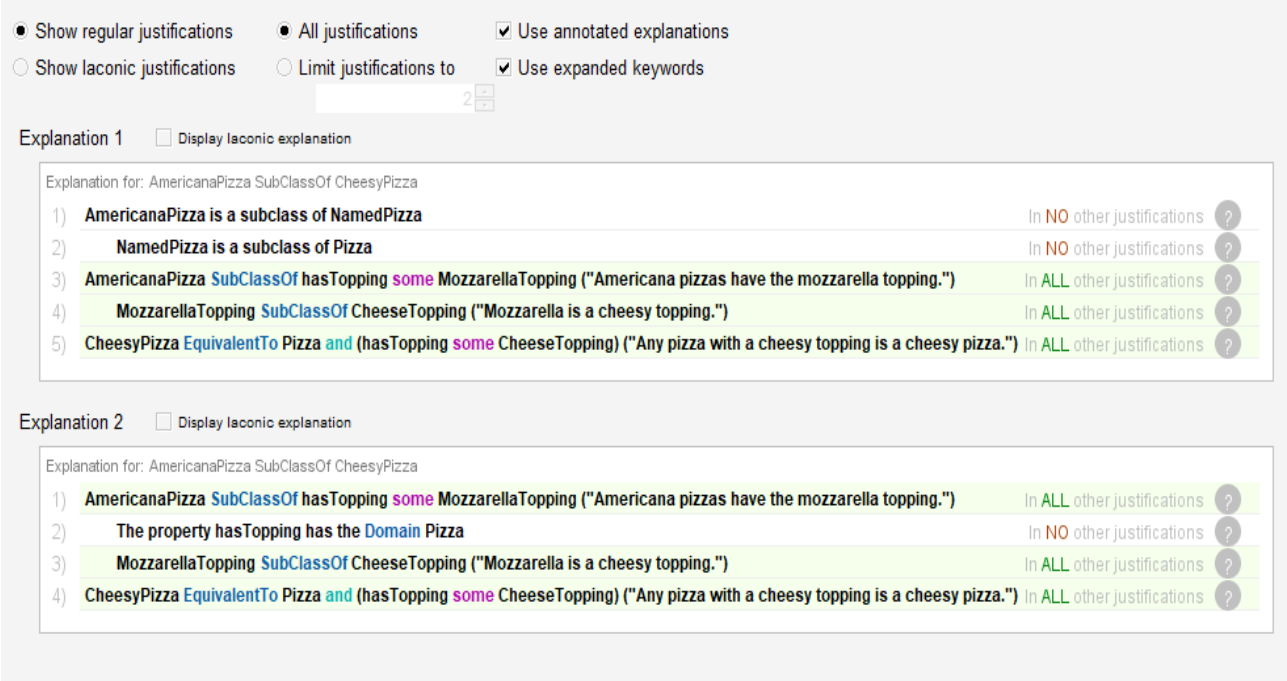

Figure 4: Output of the Explanation Workbench for an example inference, with explanatory annotations appended and keywords expanded. Note both checkboxes are selected.

7.1 Software Licence

This project is an extension of the Explanation Workbench which is published under the GNU Lesser General Public Licence 3.0. In accordance with this open source licence, this project's full source code is published under the same licence, allowing for free distribution and any modifications as long as the licence is adhered to.

8 CONCLUSIONS

The More Readable Extension to the Explanation Workbench (MRE) extends existing facilities that facilitate justification calculation for OWL ontologies in the Protégé OWL ontology editor. It provides a framework for ontology creators to define explanations for axioms and thereby allow for considerably more effective explanations.

MRE proposes an OWL Annotation Property specifically for explanations of axioms. This allows for significant future research into explanations and natural language representation of OWL and explanations.

It also provides a method for the keywords in the Manchester Syntax to be expanded to use more natural language. This is of significant value to novice users since it allows them to more easily understand description logics and the knowledge held by the ontology. It does not prohibit the more rigid and formal notation, thereby benefiting the expert users that prefer the more formal notation.

It can allow all users to receive terminological knowledge regarding the ontology with significantly greater ease and is therefore of great benefit to users wanting to familiarise themselves with a new ontology.

9 FUTURE RESEARCH

The next step in the development of this research would be to create a tool to automatically generate the explanatory annotations for axioms. This research could draw inspiration from the SWAT ontology editor and its use of OWL Simplified English to formulate the actual annotations. It might also look at Horridge's work on justifications to determine what axioms should get annotations if annotations are prioritised to the most important axioms. Associated with this, a formal definition of an explanatory annotation is a logical direction for research.

Further research can also be done to integrate the keyword expansion and the OWL Simplified English syntax. This can be integrated with user testing to evaluate the various attempts at natural language expressions for OWL and the associated explanations.

ACKNOWLEDGEMENTS

This project could not have succeeded without the support and contributions of several parties. Firstly, much gratitude is due Professor Tommie Meyer, project supervisor, for not only providing the basic framework and topic of the project but doing his best to find a project focus that is of interest to me. Thanks to Victoria

Chama for her assistance in the early research phases, and Solomon Malesa for being a willing project partner to bounce ideas from and helping with understanding the abstract theory.

REFERENCES

[1] Vicky Arnold, Nicole Clark, Philip A Collier, Stewart A Leech, and Steve G Sutton. 2006. The differential use and effect of knowledge-based system explanations in novice and expert judgment decisions. *Mis Quarterly* (2006), 79–97.

[2] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. 2017. *An Introduction to Description Logic*. Cambridge University Press. https://doi.org/10.1017/9781139025355

[3] Samantha Bail. 2013. *The justificatory structure of OWL ontologies*. Ph.D. Dissertation. The University of Manchester (United Kingdom).

[4] William J Clancey. 1983. The epistemology of a rule-based expert system – a framework for explanation. *Artificial intelligence* 20, 3 (1983), 215–251.

[5] Jasbir S Dhaliwal. 1993. *An experimental investigation of the use of explanations provided by knowledge-based systems*. Ph.D. Dissertation. University of British Columbia.

[6] Richard O Duda and Edward H Shortliffe. 1983. Expert systems research. *Science* 220, 4594 (1983), 261–268.

[7] Martha M Eining and Patrick B Dorr. 1991. The impact of expert system usage on experiential learning in an auditing setting. *Journal of Information Systems* 5, 1 (1991), 1–16.

[8] Shirley Gregor and Izak Benbasat. 1999. Explanations from intelligent systems: Theoretical foundations and implications for practice. *MIS quarterly* (1999), 497–530.

[9] Matthew Horridge. 2011. *Justification based explanation in ontologies*. Ph.D. Dissertation. The University of Manchester (United Kingdom).

[10] Matthew Horridge, Nick Drummond, John Goodwin, Alan L Rector, Robert Stevens, and Hai Wang. 2006. The Manchester OWL syntax.. In *OWLed*, Vol. 216.

[11] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, and Chris Wroe. 2004. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0. *University of Manchester* (2004).

[12] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 2008. Laconic and precise justifications in OWL. In *International semantic web conference*. Springer, 323–338.

[13] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 2009. The OWL Explanation Workbench: A toolkit for working with justifications for entailments in OWL ontologies. (2009).

[14] Izak Benbasat Ji-Ye Mao. 2000. The use of explanations in knowledge-based systems: Cognitive perspectives and a process-tracing analysis. *Journal of Management Information Systems* 17, 2 (2000), 153–179.

[15] Aditya Anand Kalyanpur. 2006. *Debugging and repair of OWL ontologies*. Ph.D. Dissertation.

[16] Deborah L. McGuinness and Peter F. Patel-Schneider. 1998. Usability Issues in Knowledge Representation Systems. In *AAAI/IAAI*.

[17] Deborah L McGuinness, Frank Van Harmelen, et al. 2004. OWL web ontology language overview. *W3C recommendation* 10, 10 (2004), 2004.

[18] Mark A Musen et al. 2015. The protégé project: a look back and a look forward. *AI matters* 1, 4 (2015), 4.

[19] Tu Nguyen. 2013. *Generating natural language explanations for entailments in ontologies*. Ph.D. Dissertation. The Open University.

[20] Richard Power. 2012. OWL Simplified English: a finite-state language for ontology editing. In *International Workshop on Controlled Natural Language*. Springer, 44–60.

[21] Thomas Roth-Berghofer and Bjorn Forcher. 2011. Improving understandability of semantic search explanations. *Int. J. Knowledge Engineering and Data Mining* 1, 3 (2011), 216–234.

[22] Rob Shearer, Boris Motik, and Ian Horrocks. 2008. HermiT: A Highly-Efficient OWL Reasoner.. In *Owled*, Vol. 432. 91.

[23] William R Swartout. 1983. XPLAIN: A system for creating and explaining expert consulting programs. *Artificial intelligence* 21, 3 (1983), 285–325.

[24] William R Swartout and Johanna D Moore. 1993. Explanation in second generation expert systems. In *Second generation expert systems*. Springer, 543–585.

[25] Randy L Teach and Edward H Shortliffe. 1981. An analysis of physician attitudes regarding computer-based clinical consultation systems. *Computers and Biomedical Research* 14, 6 (1981), 542–558.

[26] Nava Tintarev and Judith Masthoff. 2011. Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*. Springer, 479–510.

[27] Dmitry Tsarkov and Ian Horrocks. 2006. FaCT++ description logic reasoner: System description. In *International joint conference on automated reasoning*. Springer, 292–297.

[28] L Richard Ye and Paul E Johnson. 1995. The impact of explanation facilities on user acceptance of expert systems advice. *Mis Quarterly* (1995), 157–172.

APPENDIX

List of keywords in Manchester Syntax

OWL DL Concept	Manchester Syntax	Expanded Keyword
$A \subseteq B$	A SubClassOf B	A is a subclass of B
$A \subseteq B$	A Type B	A is a type of B
$A \equiv B$	A EquivalentTo B	A is equivalent to B
$A = B$	A SameAs B	A is the same as B
$r \geq i\ B$	r min i B	r no less than i B
$r \leq i\ B$	r max i B	r no more than i B
$r\ X$	r value: X	r with the value X
$r \subseteq s$	r SubPropertyOf s	r is a subproperty of s
r Domain A	r Domain A	r has the domain A
r Range A	r Range A	r has the range A
$\neg A$	not A	not A
$A \sqcap B$	A and B	A and B
$A \sqcup B$	A or B	A or B
$\exists.r\ B$	r some B	r at least one B
$\forall.r\ B$	r only B	r only B
$q = r^{-1}$	q InverseOf r	q is the inverse of r
$q = r^{-1}$	q inverse r	q is the inverse of r
Functionality	Functional: r	The property r is functional
Transitivity	Transitive: r	The property r is transitive
Reflexivity	Reflexive: r	The property r is reflexive
Symmetry	Symmetric: r	The property r is symmetric
$A \cap B = \varnothing$	DisjointClasses: A, B	The classes A and B are disjoint classes.
$r \cap s = \varnothing$	DisjointProperties: r, s	The properties r and s are disjoint properties.
$A \equiv B = \varnothing$	EquivalentClasses: A, B	The classes A and B are equivalent classes.
$r \equiv s = \varnothing$	EquivalentProperties: r, s	The properties r and s are equivalent properties.
$A \cap B = \varnothing$	A DisjointWith B	The classes A and B are disjoint classes.
$A_1 = B_1$	SameIndividual: A_1, B_1	A_1 and B_1 are the same individual.
$A_1 \neq B_1$	DifferentIndividuals: A_1, B_1	A_1 and B_1 are different individuals.
$A = \bigcup_{i \in I} A_i$	A DisjointUnionOf A1, A2	A is the disjoint union of A1 and A2.

Table 2: A table listing keywords from the Manchester Syntax and the expanded keywords used in this project.