

# Explanation in Logic-Based Reasoning Systems (EXPLOG) Project Proposal

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

Solomon Malesa  
mlssol001@myuct.ac.za  
University of Cape Town  
Rondebosch, South Africa

## CCS CONCEPTS

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

## KEYWORDS

Explanation, Description Logics, OWL

## 1 PROJECT DESCRIPTION

A logic-based reasoning system is a software system that generates conclusions that are consistent with a knowledge base (KB). This KB consists of a set of propositions that are interpreted to be true. There is an increasing trend of adopting logic-based reasoning systems as primary advisers in industries such as Medicine, e-commerce and Finance for the purposes of performing: medical diagnosis, product recommendations, and stock market decisions, respectively. Although the rate of adoption of logic-based reasoning systems has risen, the acceptance of the conclusions generated by logic-based reasoning systems is not guaranteed because the steps taken to generate conclusions are usually hidden from the user [15].

As a result, reasoning systems are perceived as black-box systems. This lack of transparency has been identified by Guidotti et al. [6] as an overwhelming drawback on the usability of reasoning systems.

The increase in computational power and decrease in costs of manufacturing computing systems has led to computing devices finding their way into everyday use in almost all areas of life. The main advantage of possessing computing devices is that they are very good at performing repetitive tasks, and more recently performing tasks that for the main part of history were left to humans, such as trading the stock market.

Taking the example of trading the stock market, the conclusions generated by a reasoning system such as *'buy company X stock at price Y'* are hardly accepted by users because there is no way of judging whether such a conclusion is correct or not. Thus, in order to decide whether to act on the reasoning system's conclusion, the system needs to provide the user with an explanation on why its conclusion is valid.

### 1.1 Explanation Definition and Types of Explanations

We define an explanation as the minimal set of propositions (also referred to as premises) that are required to be true for a conclusion

to hold. There could be more than one unique explanation for a single conclusion, hence not all explanations are equal. We begin by identifying different types of explanations identified by Swartout and Moore[19], these are: Explanations about the system behaviour, Justifications, Preferences, Domain Explanations, and Terminology definitions.

Explanations about the system's behaviour outline the systems' conclusions by paraphrasing the actions it took step by step. These types of explanations are seen in early generation explanation facilities such as the MYCIN software system. The MYCIN software system was developed in the early 1970s at Stanford University, for the purposes of diagnosing bacterial infections. It produced its conclusions using a technique known as backward chaining whereby the system starts with a list of goals and works backward to determine if the KB contains the required premises.

According to Horridge et al. [10], Justification explanations are: "a minimal subset of the ontology that is sufficient for an entailment to hold". An ontology<sup>1</sup> is the KB of the reasoning system and an entailment is a conclusion. Furthermore, Horridge et al. [10] subdivides Justifications into Laconic Justifications and Precise Justification:

"Laconic justifications only consist of axioms that do not contain any superfluous *parts*. Precise justifications can be derived from laconic justifications and are characterised by the fact that they consist of flat, small axioms, which facilitate the generation of semantically minimal repairs"

Explanations of this kind provide the rationale behind the reasoning system's conclusions [19].

Preferences are explanations that provide the rationale behind why certain explanations are more valuable than others. These sorts of explanations provide invaluable insights when comparing the quality of explanations. Domain explanations define the problem domain itself [19]. An example of explanations of this kind are the explanations generated by the Digitalis Advisor, a program designed to give advice to physicians about therapeutic treatments and recommendations. The last explanation identified by Swartout and Moore is the Terminology Definition Explanation. Terminology definitions are explanations that are focused on the definitions of the terminology used by a KBS [19]. For instance, the Digitalis

<sup>1</sup>Ontology is also a study concerned with how to efficiently represent knowledge by selecting appropriate formal names for concepts, defining categories, properties and relations between the concepts, e.t.c.

Advisor may have to answer a query such as “What is a compounded drug?”.

## 1.2 Characteristics of Explanations

In order to evaluate the quality of explanations, we must define benchmarks which form a criteria that defines what makes an explanation a good explanation. Swartout and Moore [19] have identified the following aspects as the elements of a good explanation: Fidelity, Understandability, Sufficiency, Low Construction Overhead and Efficiency.

**Fidelity** focuses on how well does the explanation represent what the reasoning system really does.

**Understandability** focuses on whether the user understands that the reasoning system generates. This aspect 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.

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

**Low Construction Overhead.** This focuses on the difficulty of generating explanations. Swartout and Moore [19] suggest that it should not be more difficult to construct an explanation than constructing a reasoning system.

**Efficiency** focuses on the impact of generating an explanation on the run time of reasoning systems.

The aforementioned elements of a good explanations are widely accepted as a heuristics benchmark which is used to qualitatively measure the usability of explanations.

## 1.3 Challenges in Current Explanations

The scarcity of explanations in reasoning systems suggests that the area concerned with generating explanations or augmenting reasoning systems with explanation facilities is either relative new or is riddled with challenges. Nguyen [17] and Horridge et al. [10] claim that deriving explanations from a set of premises is not an easy task, especially when it is a large explanation. Furthermore, they claim that the explanations generated by current reasoning systems are too difficult to understand so much that even experts of this field struggle to understand them.

Swartout and Moore [12] have identified the following key challenges of explanation facilities that are still pervasive to this day:

**1.3.1 Narrowness.** . Explanation systems seem to allow a limited variety of queries.

**1.3.2 Inflexible.** . The presentation of the explanations is limited to text format and does not cater for other formats such as graphical forms.

**1.3.3 Insensitive.** . Many explanation facilities cannot be tailored to users’ unique requirements.

**1.3.4 Unresponsive.** . Many explanation facilities are not able to elaborate their explanations or answer follow up questions.

**1.3.5 Inextensible.** . Users cannot modify or extend many existing explanation facilities.

In this project, we focus on increasing the user acceptance of reasoning systems by generating explanations for conclusions made by the logic-based reasoning systems. As stated in the project brief, this project comprises two aspects: a theoretical component in which the formal aspects of explanations are dealt with and a practical component in which a tool for explanations is developed.

## 1.4 Theoretical Aspect

The scope of the theoretical component is flexible but will focus on generating justification explanations. As stated in the background section, these are minimal sets of premises that required to be true for a conclusion to hold. At a high-level view, the theoretical component will focus on defining what are justification explanations, how to generate explanations, and how to convert explanations into a natural language.

## 1.5 Practical Implementation

The practical aspect of the project will, at a high level, consist of improving and extending current explanation tools for Protégé to provide explanations that are more readable to humans. In particular, the implementation aims to introduce an explanation template (created by the ontology creator) to objects in an OWL ontology that can be used for explanation. Current explanation tools for Protégé displays only the subclasses used in a particular reasoning.

For example, say there is an ontology with two classes, *Wind* and *Direction*. For the purposes of this example, *Direction* is enumerated as  $\{North, East, South, West\}$ . Every instance of *Wind* has the property *hasDirection.D*. *hasDirection.D* is a relation from *Wind* to *Direction*. Say there is another class, *SoutherlyWind*, defined as everything that is  $Wind \sqcap \forall hasDirection.South$ .

Say we have an individual  $W1$ .  $W1 \sqsubseteq Wind$  and only *hasDirection.South*. It is clear that  $W1 \sqsubseteq SoutherlyWind$ , and the Protégé reasoner will infer this. If an explanation is requested, the current Protégé Explanation Workbench will provide the statements  $W1 \sqsubseteq Wind$ ,  $W1 \forall hasDirection.South$ , and  $SoutherlyWind \equiv Wind \sqcap \forall hasDirection.South$ . Since it is rather intuitive from the class names that  $SoutherlyWind \sqsubseteq Wind$ , the central statement of  $W1 \forall hasDirection.South$  will be of greatest prominence for most users.

The practical aspect of the project hopes to enable an explanation template, such that when a relation is defined, the template is defined as well. In this example, the template for the relation *hasDirection.D* could be  $\langle Class \rangle \langle Domain \rangle has [Qualifier] the direction \langle Range \rangle$ . Or for the specific individual  $W1$ , the statement  $W1 \forall hasDirection.South$  will be  $Wind W1 has only the direction South$ . The  $\langle Class \rangle$  and  $\langle Domain \rangle$  are determined by the individual, the  $[Qualifier]$  is determined by the forall or existential qualifier on the relation, and the  $\langle Range \rangle$  is determined by the value of the relation.

These templates are easily implemented in a modular fashion. The initial focus will be on providing the template functionality for relations only, and if there is time available, expand it to include set subsumption and object properties as well.

## 2 MOTIVATION FOR PROJECT

Explanations are considered a key part of knowledge-based systems used by professionals. [11] Including explanations in a KB system leads to a marked increase in user acceptance of the system's recommendations. [18] This closer adherence to system recommendations lead to users having more accurate performance in a shorter time. ([1], [4], [5], [20])

The current Protégé tool has an explanation tool incorporated. However, these explanations are difficult to read and near impossible to understand if the user did not design the ontology themselves, since it is very formulaic and uses mathematical symbols only. There have been several papers establishing explanations for OWL ontologies. ([3], [13], [16]) This includes attempts at generating explanations in forms closer to natural language. [16]

Despite the volume of research regarding usable explanations for OWL ontologies, there are no actual implementations available for the Protégé tool. By leaving the final explanation design up to the ontology creator, the explanations are more accurate and more useful for the users of the ontology.

## 3 PROBLEM STATEMENT

In the project description section we discussed why end-users hardly accept explanations generated by reasoning systems. Furthermore, we also mentioned that our ultimate goal is to increase the acceptability of explanations and/or increase the usage of reasoning systems. Given that this project comprises two components (a practical component and a theoretical component), each component has its own research question and objectives.

The aim of the project is to explore explanation in logical programming, with a deeper focus for each of the group members on the theoretical and practical aspects.

### 3.1 Theoretical Aspect

The theoretical component of this project aims to equip Ontology Engineers and other domain experts with a comprehensive theoretical knowledge that is required to build explanation facilities for reasoning systems. We curtail the theoretical component's scope by only using description logics, specifically we focus on generating explanations for conclusions drawn by subsumption. To achieve this goal we answer the following research questions:

- (1) How to convert the execution trace of a program that produces a conclusion into an explanation?
- (2) How to explain conclusions that have been reached through a long series of deductions?
- (3) How to explain negative conclusions<sup>2</sup>?
- (4) Given a set of unique explanations for a single conclusion, how to determine and select the best explanation?

<sup>2</sup>We define negative conclusions as a type of conclusion that does not have an explanation.

- (5) Given an explanation which is in a description logics syntax form, how to convert it into a more friendlier English language form?

### 3.2 Practical Aspect

In particular, the aim of the practical aspect is to provide a plugin for the Protégé tool that provides explanations that are more readable to humans and less reliant on an inherent understanding of the ontology to understand. The end goal is an extension of the current explanation tool integrated into Protégé to allow for explanation templates. Initially, the scope will be limited to only relations, with expansions possible if there is time for it.

## 4 PROCEDURES AND METHODS

### 4.1 Theoretical Aspect

The first thing we need to do is acclimatise ourselves with Basic Description Logics. This means reading an introductory book on the subject. At the time of writing this paper, the book of interest is: "An Introduction to Description Logic" [2]. From this book and other relevant materials such as Horridge's [10] paper on '*Laconic and Precise Justifications in OWL*' we will characterise and formally define explanations in terms of description logics. This includes looking at the types explanations and other traits such as those mentioned in section 1.2.

In order to address the research questions posed in section 2.1, the following steps will be taken<sup>3</sup>:

- (1) We assume that the program can be expressed using description logics. Therefore, the first step required to answer this question is to convert the execution trace if a program into an equivalent description logics ordered set of steps. It is feasible to view an execution trace that produces a conclusion as a proof of how the program produced the conclusion, this is justified by fact that if you recall from the project description, such a proof is classified as an explanation about the program's behaviour that produces a conclusion. Thus, we will treat every proof as an explanation.
- (2) One approach that could be used to address this question is to find a way of decomposing long serial proofs into atomic segments<sup>4</sup> of the proof. We then specify the fidelity of the explanation and determine which segments need to be joined and how they should be joined to return a much more palatable explanation. At the moment of writing this paper, this has been the only reasonable way of answering this question. Further research will be done to determine if there are other better methods of answering this question.
- (3) The notorious '*proof-by-contradiction*' is used to provide a counter-conclusion such that if both the negative conclusion and the counter-conclusion are true, then some proposition (preferably a propositional atom) drawn from the KB has to be false. We choose this technique because we are familiar with it and it is prominent in the field of computer science. It is powerful enough to prove and disprove theorems, which

<sup>3</sup>Note that the number of the steps corresponds to the research questions in section 2.1.

<sup>4</sup>These are synonymous with propositional atoms.

in our case, we consider the theorem to be the reasoning system's conclusion and the execution trace to be the proof of the theorem. Thus, by example, if a false theorem is assumed to be true then it suffices to show that some mathematical axiom will then have to be false.

- (4) Given that the quality of an explanation is highly dependent on the user's experience, we will investigate how the elements that constitute a good explanation (as discussed in section 1.2) affect the user's reception to the explanation. This will require sifting and searching through multiple peer-reviewed conference papers for empirical data on how domain experts perceive explanations with respect to the aforementioned elements.
- (5) This will require us to acclimatize ourselves with natural language processing techniques. This will be done by reading Nguyen's Chapter 6 Verbalization for Deduction Rules [16] and a paper that deals with improving readability of OWL Ontologies [9]. Our current strategy is to develop templates by associating each description logics operation with an equivalent English language syntax and leaving the operands as variables which can be replaced by the relevant object of the KB.

All the methods followed in the theoretical component of this project will be deemed successful if a formal, correct and sufficient mathematical description is obtained at the end of each procedure. Because this part of the project is theoretical, the quality of the mathematical descriptions generated is subjective to personal preference. Therefore, we will consult with our supervisor to help us determine other ways of evaluating the quality of our theoretical results.

## 4.2 Practical Aspect

For implementing the extension, the source code for both the Protégé Editor and the Explanation Workbench Plugin is freely available under open-source licences. There is an API for developers to utilise when developing plugins for Protégé. There are also developer tutorials available on the Protégé Wiki.

Protégé is written in the Java programming language, and there are guidelines on setting up Java IDEs such as Eclipse to function optimally for developing plugins for Protégé. To extend the plugin, we will familiarise ourselves with the functioning of the existing explanation tool and Protégé from a programming perspective.

Once familiar, we will create the framework for explanation templates for relationships and test the framework using an example ontology. A few rounds of testing, debugging, and fine-tuning using the example will follow. Once it works perfectly for one ontology, other ontologies will be used for testing to ensure the framework is generalisable.

By this time, we should have a good idea what it takes to create a generalisable framework for explanation templates. If there is enough time left, the scope will be expanded to include explanation templates for subsumption of classes. This will follow the same pattern of testing and improvement as for the relationship templates.

## 5 ETHICAL, PROFESSIONAL, AND LEGAL ISSUES

There are no ethical issues inherent to this project, given that it is mostly theoretical and does not involve testers at all. The project and the intellectual property thereof will be dispensed as per the University of Cape Town's policies. The practical implementation will be an extension of the Protégé Explanation Workbench that is published under the Apache Licence Version 2.0 and GNU Lesser General Public Licence 3.0 licences. In accordance with these open-source licences, any modifications made will be published under the same licences, with full source code available and permission for free access and any modifications as long as the licences are adhered to.

## 6 RELATED WORK

### 6.1 Theoretical Aspect

McGuinness [14] was one of the earliest people to publish a dissertation on generating explanations for reasoning systems using description logics. Her work provides 'classical' techniques that can be used to answer the first three research questions in section answers to our first established the traditional techniques used to address the first three research questions we posed in section 2.1. She also identifies one of the challenges in this area for the first research question that many procedural codes are being optimised so much that the program trace does not suffice as a form of an explanation about the reasoning system's behaviour.

Some of the most illuminating related work is produced by Horridge et al. [9], in which they attempt to address the fifth research question by building a new OWL Syntax, called The Manchester OWL Syntax. They designed this syntax as a "response to a demand from non Description Logic users for less logician like syntax". Although they do not produce explanations in the form everyday English language, the syntax uses natural language keywords to alleviate the cognitive load required from the user.

Another attempt at the fifth research question is posed by Hart et al. [7], they have designed an OWL plugin called Rabbit. It provides "easy-to-construct constrained English sentences that can be directly translated into OWL". Furthermore, Hart et al. [7] point out that domain experts are slow to adopt OWL.

Although McGuinness' [14] work is not recent, some of her ideas are worth looking into because they are still in use today. For instance, to address the second research question in section 2.1, she suggests that objects of a KB must be tagged with special identifiers to indicate that they are of a special interest. Therefore if an explanation happens to be very long, the user will be able to filter out information according to the required fidelity/granularity.

She summarises her contributions [14] as follows:

- "1. A proof-theoretic foundation for explanations of subsumption and membership in description logics, together with techniques for presenting explanations in manageable chunks. 2. A method for explaining

nonsubsumption by generating counter-examples using the existing implementation of the classic system and its explanation component. 3. An object oriented method of pruning object and explanation presentations”

Nguyen [16] is another contributor in this field. She “proposed a method of measuring the cognitive difficulty of single-step inferences in OWL’. This was done by establishing an understandability index which based on a probabilistic model. However, her study is riddled with many limitations such as low rule coverage which means that at some point her system will fail to produce an explanation that requires rules that have not been covered.

Input	<p><b>Entailment:</b> <math>Person \sqsubseteq Movie</math> (Every person is a movie)</p> <p><b>Justification:</b></p> <ol style="list-style-type: none"> <li><math>GoodMovie \equiv \forall hasRating.FourStarRating</math> (A good movie is anything that has only four stars as ratings)</li> <li><math>Domain(hasRating, Movie)</math> (Anything that has a rating is a movie)</li> <li><math>GoodMovie \sqsubseteq StarRatedMovie</math> (Every good movie is a star-rated movie)</li> <li><math>StarRatedMovie \sqsubseteq Movie</math> (Every star-rated movie is a movie)</li> </ol>
Output	<p><b>Explanation:</b></p> <p>The statement “Every person is a movie” follows because the ontology implies that “Everything is a movie” (a).</p> <p>Statement (a) follows because:</p> <ul style="list-style-type: none"> <li>- anything that has as rating something is a movie (from axiom 2), and</li> <li>- everything that has no rating at all is a movie (b).</li> </ul> <p>Statement (b) follows because:</p> <ul style="list-style-type: none"> <li>- everything that has no rating at all is a good movie (c), and</li> <li>- every good movie is a movie (d).</li> </ul> <p>Statement (c) follows because axiom 1 in the justification actually means that “a good movie is anything that has no rating at all, or has only four stars as ratings”.</p> <p>Statement (d) follows because:</p> <ul style="list-style-type: none"> <li>- every good movie is a star rated movie (from axiom 3), and</li> <li>- every star rated movie is a movie (from axiom 4).</li> </ul>

**Figure 1: A table illustrating of an input entailment-justification pair generated by Nguyen’s explanation facility [16]**

## 7 EXPECTED OUTCOMES

### 7.1 Theoretical Aspect

We expect to provide a firm theoretical framework for generating good explanations. As discussed in the procedures and methods section, the success of this framework will be dependent on the target users’ feedback. In our case, we will consult with our supervisor through out the project to determine the quality of our work. We anticipate facing challenges similar to those that the related work has come across. For instance, Bail [3] indicates that it neither clearly defined nor easy process to measure the understandability of Ontologies. On the other hand, Horridge [8] points out that establishing relationship between syntax and semantics was the most challenging aspect of his dissertation. Both authors, Bail and Horridge [3, 8], claim that as the number of justifications for conclusions grows, it proves difficult to select laconic justifications.

We expect our framework will contribute the following:

- Provide a mechanism for tuning the fidelity of explanations
- Increase the runtime efficiency of generating explanations
- Improve the understandability of explanations
- Provide methods of reducing long chains of explanations into small understandable and manageable chains of explanations

Finally, we anticipate that the scope of our project may change during the period of project implementation.

### 7.2 Practical Aspect

We expect the Explanation Workbench to provide at least some explanations in a form that is human-readable and comprehensible to users who did not design the ontology. Specifically, the practical aspect of the project will be judged a success if it provides explanations in a more understandable and natural form.

## 8 PROJECT PLAN

### 8.1 Project Timeline and Major Milestones

A Gantt chart laying out the projected timeline can be found as in the appendix. The timeline was constructed by following the supplied project milestones as a guideline.

### 8.2 Resources Required

For the practical part, the resources required are an IDE capable of developing a plugin for Protégé, a computer that can run Protégé, and the source code of Protégé and the explanation workbench. The source code is freely available under open-source licences.

### 8.3 Risk Matrix

Table 1 in the appendix show the risks associated with the project laid out in a risk matrix. Each risk is evaluated by its probability and the impact if it occurs. Strategies for mitigation, monitoring, and management of the risk are then proposed.

### 8.4 Work Allocation

Solomon Malesa will be responsible for the theoretical aspect of the project. He will formally define explanations, describe how to generate them and how to translate explanations into a natural language. Cilliers Pretorius will be responsible for the practical aspect of implementing an extension to the current explanation tools in Protégé.

## 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] 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.
- [5] Shirley Gregor and Izak Benbasat. 1999. Explanations from intelligent systems: Theoretical foundations and implications for practice. *MIS quarterly* (1999), 497–530.
- [6] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local Rule-Based Explanations of Black Box Decision Systems. *CoRR* abs/1805.10820 (2018). arXiv:1805.10820 <http://arxiv.org/abs/1805.10820>
- [7] Glen Hart, Martina Johnson, and Catherine Dolbear. 2008. Rabbit: Developing a Control Natural Language for Authoring Ontologies. In *The Semantic Web: Research and Applications*, Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 348–360.
- [8] Matthew Horridge. 2011. Justification based explanation in ontologies.
- [9] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai Wang. 2006. The Manchester OWL syntax. *Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006)*.
- [10] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 2008. Laconic and Precise Justifications in OWL. In *The Semantic Web - ISWC 2008*, Amit Sheth, Steffen Staab,

- Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 323–338.
- [11] 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.
  - [12] William Swartout Johanna Moore. 1989. Explanation in expert systems: A survey. (01 1989).
  - [13] Aditya Kalyanpur. 2006. *Debugging and Repair of Owl Ontologies*. Ph.D. Dissertation. College Park, MD, USA. Advisor(s) Hendler, James. AAI3222483.
  - [14] Deborah McGuinness. 1996. Explaining Reasoning In Description Logics. (11 1996).
  - [15] Deborah L. McGuinness and Peter F. Patel-Schneider. 1998. Usability Issues in Knowledge Representation Systems. In *AAAI/IAAI*.
  - [16] Tu Nguyen. 2013. *Generating natural language explanations for entailments in ontologies*. Ph.D. Dissertation. The Open University.
  - [17] Tu Anh T. Nguyen, Richard Power, Paul Piwek, and Sandra Williams. 2012. Planning Accessible Explanations for Entailments in OWL Ontologies. In *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*. Association for Computational Linguistics, Utica, IL, 110–114. <https://www.aclweb.org/anthology/W12-1518>
  - [18] 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.
  - [19] William Swartout and Johanna Moore. 1993. Explanation in Second Generation Expert Systems. 543–585. [https://doi.org/10.1007/978-3-642-77927-5\\_24](https://doi.org/10.1007/978-3-642-77927-5_24)
  - [20] 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.

## 9 APPENDICES