Generating Natural Language Explanations For Entailments In Ontologies

Thesis

GENERATING NATURAL LANGUAGE EXPLANATIONS FOR ENTAILMENTS IN ONTOLOGIES

A THESIS SUBMITTED TO THE OPEN UNIVERSIY (UNITED KINGDOM)
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF MATHEMATICS, COMPUTING & TECHNOLOGY

2013

by
Tu Anh T. Nguyen
Department of Computing

Contents

List of Figures

List of Tables

Abstract

Building an error-free and high-quality ontology in OWL (Web Ontology Language)—the latest standard ontology language endorsed by the World Wide Web Consortium—is not an easy task for domain experts, who usually have limited knowledge of OWL and logic. One sign of an erroneous ontology is the occurrence of undesired *inferences* (or *entailments*), often caused by interactions among (apparently innocuous) axioms within the ontology. This suggests the need for a tool that allows developers to inspect why such an entailment follows from the ontology in order to debug and repair it.

This thesis aims to address the above problem by advancing knowledge and techniques in generating explanations for entailments in OWL ontologies. We build on earlier work on identifying minimal subsets of the ontology from which an entailment can be drawn—known technically as *justifications*. Our main focus is on planning (at a logical level) an explanation that links a justification (premises) to its entailment (conclusion); we also consider how best to express the explanation in English. Among other innovations, we propose a method for assessing the understandability of explanations, so that the easiest can be selected from a set of alternatives.

Our findings make a theoretical contribution to Natural Language Generation and Knowledge Representation. They could also play a practical role in improving the explanation facilities in ontology development tools, considering especially the requirements of users who are not expert in OWL.

Acknowledgements

First and foremost, I would like to thank my supervisors: Richard Power, Paul Piwek, and Sandra Williams. I would never have completed a PhD without them, and I could not have asked for better supervisors. They have always there for me to provide encouragement and helpful feedback on my research as well as skill development (especially writing).

I would like to thank Tru Hoang Cao, who was largely responsible for introducing me to Semantic Web and work in this area; and Robert Stevens, who introduced me to justifications (for entailments) and their difficulty in understanding.

A very special thank you to Matthew Horridge, whose PhD research has inspired me a lot, for providing me the programs I need for this research.

I would like to thank other members of the SWAT (Semantic Web Authoring Tool) project and the NLG (Natural Language Generation) group at the Open University—in particular Donia Scott, Alan Rector, Allan Third, Fennie Liang, Svetlana Stoyanchev, and Brian Pluss—and Bijan Parsia for helpful discussions and comments on my research and papers. Thanks to Marian Petre for being such a kind third party monitor.

I dedicate this thesis to my family for their unconditional love and constant support, and my husband, Hop Minh Nguyen, for always being there for me and putting up with me while I was doing this PhD.

Chapter 1

Introduction

In computer science, an *ontology* is defined as an engineering artifact that introduces vocabulary describing the relationships between concepts and objects in some domain of interest, and explicitly specifies the intended meaning of that vocabulary [Hor08]. An ontology is often encoded into a machine-processable form by using a formal logic-based language called an *ontology language*. The latest standard ontology language is OWL (Web Ontology Language), endorsed by the W3C (World Wide Web Consortium) in 2004. Since then OWL has become widespread in many domains. A number of important ontologies have been built in OWL to standardize the representation of knowledge in different fields; examples are the gene ontology [ABB+00] in bio-informatics, the NCI (National Cancer Institute) Thesaurus [GFH+03], and the SNOMED ontology [SCC97] in health-care.

Building a high-quality ontology, such as one of those mentioned above, is not an easy task, especially for people with limited knowledge of OWL and logic [Pul96, ADS+07]. In addition to obstacles caused by the opacity of OWL standard syntaxes[1] [ST04], and difficulties in understanding the logic-based semantics of OWL axioms [RDH+04], there remains the more subtle problem of undesired *inferences* (or *entailments*), often caused by interactions among innocuous-looking axioms in the ontology. The difference between axioms and entailments is that whereas axioms are statements asserted by a user into

[1] The primary OWL syntax is RDF/XML (Resource Description Framework/eXtensible Markup Language), which is an XML syntax for encoding RDF graphs in a serialisation format [RDF04].

Table 1.1: An example of explanations generated by the explanation system for the input entailment-justification pair

Input	**Entailment:** $Person \sqsubseteq Movie$ (Every person is a movie) **Justification:** 1. $GoodMovie \equiv \forall hasRating.FourStarRating$ (A good movie is anything that has only four stars as ratings) 2. $Domain(hasRating, Movie)$ (Anything that has a rating is a movie) 3. $GoodMovie \sqsubseteq StarRatedMovie$ (Every good movie is a star-rated movie) 4. $StarRatedMovie \sqsubseteq Movie$ (Every star-rated movie is a movie)
Output	**Explanation:** The statement "Every person is a movie" follows because the ontology implies that "Everything is a movie" (a). Statement (a) follows because: - anything that has as rating something is a movie (from axiom 2), and - everything that has no rating at all is a movie (b). Statement (b) follows because: - everything that has no rating at all is a good movie (c), and - every good movie is a movie (d). 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". Statement (d) follows because: - every good movie is a star rated movie (from axiom 3), and - every star rated movie is a movie (from axiom 4).

an ontology, entailments are statements that logically follow from axioms. Since OWL provides a highly expressive set of axioms, it is quite common for apparently innocuous OWL axioms, when combined together, to lead to absurd conclusions such as "Everything is a person", "Nothing is a person", or indeed "Every person is a movie". Any such entailment signals that the ontology needs to be debugged.

When debugging an undesired entailment, many users, especially those with limited knowledge of OWL and logic, will need more information in order to make the necessary corrections: they need to understand *why* the undesired entailment was drawn, before they can start to repair it. In such cases, providing a natural language explanation of why the entailment follows from the ontology, such as the one in Table 1.1 for the (obviously absurd) entailment "Every person is a movie", would be of great help to users. Automated reasoners [TH06, SPG+07], as will be explained in Chapter 2, can compute entailments efficiently, but provide no useful information that helps generate such an explanation. Earlier work on identifying minimal subsets of the ontology from which an entailment can be drawn—known technically as *justifications*[2] [Kal06]—help pinpoint which axioms

[2]The minimality requirement here means that if any axiom is removed from a justification, the entailment will no longer be inferable.

in the ontology are responsible for the entailment, and hence, provide a good basis for explaining it. However, since there may be multiple justifications for an entailment, it is unknown which of several justifications provides the best basis for explaining it.

Providing a justification for an entailment, understanding how to get from the justification (premises) to the entailment (conclusion) is not always easy, especially when the size of the justification is large. Additionally, Horridge et al. [HPS09b, Hor11] showed that there exist naturally occurring justifications from OWL ontologies that are so difficult that even OWL experts are unable to understand without further explanation; the justification shown in Table 1.1 is an example of such cases. To help users with understanding a justification, they proposed a method for computing intermediate statements (or *lemmas*) representing steps in inferring the entailment from the justification. However, this method introduces lemmas only in some fixed subsumption forms, and in a somewhat random manner, so it is not guaranteed to provide lemmas that facilitate human understanding. Moreover, among various possible lemmas that can be introduced into a justification, it is still unknown which one provides the best subdivision of the inference into multiple steps.

1.1 Research Problem and Methodological Approach

The scope of this thesis is restricted to explanations for non-trivial *subsumption entailments* between two class names—i.e., entailments of the forms $\top \sqsubseteq A$ (Everything is an A), $A \sqsubseteq \bot$ (Nothing is an A), and $A \sqsubseteq B$ (Every A is a B), where A and B are class names—as they are the main inference types in description logics[3]. It aims to address the following three research questions:

1. How can we find deduction patterns that are suitable for single inference steps in an explanation (considering both frequency and level of understandability)?

2. Given a set of alternative multi-step explanations, how can we provide an empirically grounded criterion for deciding which is most suitable—i.e., which would be understood best by our target users, assumed to be non-logicians?

3. Having determined the logical structure of an explanation, how can we most clearly

[3]Explanations for other entailment types are topics for future work.

express it in English through appropriate ordering, verbalisation, and extra elucidation?

In order to address the first question, an empirical study is conducted to thoroughly examine justifications for entailments computed from large a corpus of published real world OWL ontologies. The main focus of this study is on determining frequently occurring *deduction patterns* in the corpus, and more importantly, how to formulate *deduction rules* as generic representatives of basic inferences in OWL, especially those that lead to a contradiction or an undesired subsumption, from these patterns. Another study is conducted to empirically assess the *level of understandability* of the rules to non-logicians. These rules are then employed to generate lemmas for a justification. Lemmas introduced in this way are user-oriented as they subdivide the inference into steps that are neither too trivial nor too complex to be understood by most users.

Given a justification and a set of deduction rules, there may exist multiple ways to introduce lemmas into the justification, and so multiple explanations, some of which may be easier to follow than others. Hence, the capability of assessing the understandability of explanations so that the easiest can be selected from a set of alternatives would be of great help to end-users. Additionally, when multiple justifications for an entailment are found, this would enable the sorting of explanations in order of decreasing understandability, which is also useful for end-users. To address this problem (which is raised in the second research question), a model capable of predicting the understandability of a multi-step explanation is proposed. This model is based on assessment of understandability of our rules—which are in fact single-step inferences in OWL.

To address the third question, we first investigate how best to express a single-step inference in English—in other words, identifying which of various possible verbalisations for a deduction rule is most understandable. This involves the identification of best verbalisations for OWL axioms, and more importantly, the determination of the best ordering for premises in the rule as well as the arguments within each premise. We rely on existing theoretical insights from the psychology of reasoning to identify the best verbalisations for most rules. When prior work provides no guidance, we use an empirical test to find out which verbalisations are understood best by non-logicians. For rules that are relatively difficult to understand (identified through one of our prior studies), we seek suitable

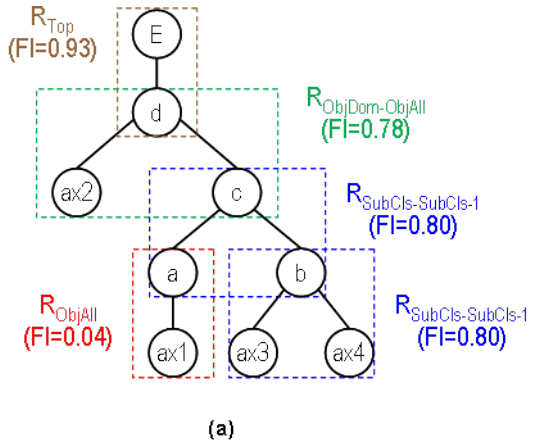


Figure 1.1: The underlying proof tree of the explanation in Table 1.1. The labels 'R_{Top}', 'R_{ObjAll}', etc. refer to deduction rules named 'Top', 'ObjAll' etc. listed in Table 4.3. 'FI' values represent understandability indexes (or *Facility Indexes*), indicating how easy it is to understand the rules, with values ranging from 0.0 (hardest) to 1.0 (easiest).

strategies for further elucidating the inference. The strategies are then compared empirically to determine the most suitable for each rule.

Our aim in answering the three above-mentioned research questions is to develop a NLG (Natural Language Generation) system capable of generating accessible explanations, in English, of why an entailment follows from an OWL ontology. This system takes as input an entailment, and outputs a number of English explanations, one for each justification of the entailment. To produce such explanations, the system starts from justifications, which can be computed efficiently by a number of available algorithms [KPHS07, Sun09]. It then constructs *proof trees* for each justification in which the root node is the entailment, the terminal nodes are the axioms in the justification, and other nodes are lemmas. If the deduction rules result in multiple proof trees for a justification, only the most understandable is selected. If there are multiple justifications for the entailment, the selected proof trees are sorted in order of decreasing understandability. Finally, the system plans and generates an English explanation from each proof tree. The architecture of the system is summarised in Figure 1.2. Its output explanation is more accessible than one based on the justification alone in two ways: (1) it replaces a single complex inference with a number of simpler inference steps, and (2) inference steps are expressed in English, and further elucidated when necessary.

As an example, Table 1.1 shows an explanation that our system generates for the entailment "Every person is a movie" based on the associated justification in the table. The underlying proof tree is shown in Figure 1.1. The key to understanding this explanation

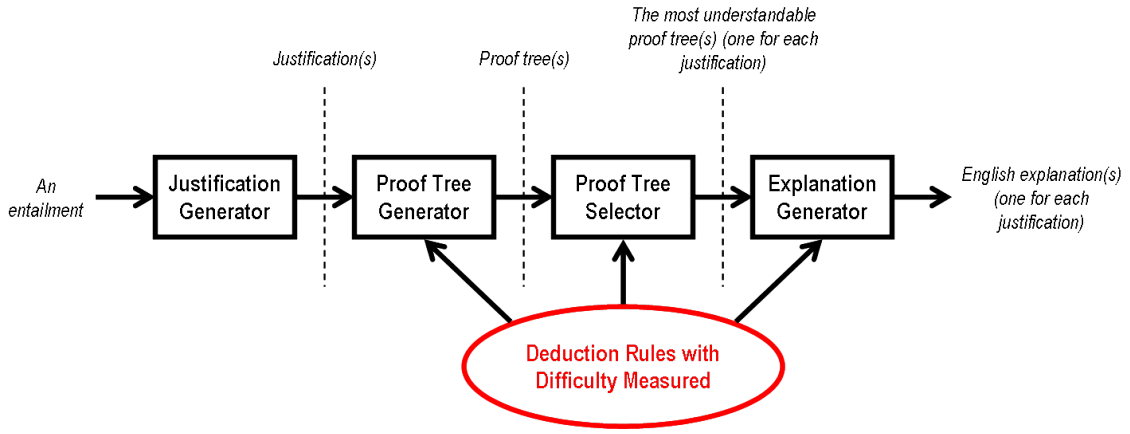


Figure 1.2: Architecture of the explanation system

lies in the inference step from axiom 1 to statement (c), which is an example of a difficult step in need of "further elucidation". Such an explanation can help identify which axiom or axioms in the ontology are problematic, and hence, need to be corrected. In this example, we can identify the problematic axiom which causes the absurd entailment by tracing from the entailment to a sequence of absurd lemmas, including "Everything is a movie", "Everything that has no rating at all is a movie", and "Everything that has no rating at all is a good movie", finally reaching the mistaken axiom "A good movie is anything that has only four stars as ratings".

1.2 Contributions

In summary, this thesis aims to advance knowledge and techniques in generating explanations for entailments in OWL ontologies, by addressing the three research questions mentioned above. In doing this, it makes the following contributions:

1. It provides a list of frequently occurring *deduction patterns* in published real world OWL ontologies, and a list of *deduction rules* as generic representatives of basic OWL inferences at the logical level, especially inferences that lead to a contradiction or an undesired subsumption.

2. It proposes a method of computing user-oriented *lemmas* for a justification to subdivide the original inference into multiple inference steps, each of which is neither too trivial nor too complex to be understood by most users, by using the set of deduction rules.

3. It implements an algorithm for constructing *proof trees* (or explanation plans) from a justification, and carries out a thorough evaluation which shows that the algorithm works relatively well in computing proof trees from justifications with ten or fewer axioms in published real world OWL ontologies.

4. It proposes the innovation of using empirical studies of understandability to guide the selection of which inference steps stand in need of further elucidation, and which proof tree would be understood best by users. Specifically, it proposes a method for empirically measuring the understandability of deduction rules (single-step inferences in OWL) to people who are not OWL experts. Additionally, it proposes an empirically grounded model capable of predicting the understandability of proof trees (multi-step inferences in OWL), and a methodology for extending and refining the model. These findings are definite advances over previous work, and employ a clear and rigorous methodology that could be applied to cover a wider range of OWL inferences, and could be generalised to any other application in which the aim is to generate understandable logical explanations.

5. It implements an NLG system that can automatically generate English explanations for subsumption entailments in consistent OWL ontologies.

1.3 Plan of the Thesis

We begin by providing the necessary background on reasoning in description logic and the computation of justifications for entailments of ontologies (Chapter 2). This is followed by a review of prior work on automatic generation of explanations for entailments and explanations for mathematical theorems (Chapter 3).

Thereafter, the components of our own explanation generator are described: deduction rules based on a corpus study of justifications in published real world OWL ontologies (Chapter 4); an algorithm for constructing candidate proof trees (Chapter 5); and mappings of deduction rules (single-step inferences) to English templates for explanations (Chapter 6).

The components mentioned above provide a means of generating many candidate explanations, but not of determining which is most suitable; this issue is addressed by the next two

chapters, which investigate empirically which inference steps are understood most easily, both singly and in combination (Chapter 7), and which strategies are most successful in explaining particularly difficult inference steps (Chapter 8).

Finally, we show how all these ideas have been implemented in a system that generates explanations for entailments (Chapter 9), and present a preliminary evaluation of this system (Chapter 10); Chapter 11 concludes.

1.4 Published Work

The work embodied in this thesis is supported by several conference and workshop publications, and a technical report:

[NPPW13] Tu Anh T. Nguyen, Richard Power, Paul Piwek, and Sandra Williams. 2013. Predicting the Understandability of OWL Inferences. In *Extended Semantic Web Conference (ESWC 2013)*, volume 7882, pages 109–123.

[NPPW12a] Tu Anh T. Nguyen, Richard Power, Paul Piwek, and Sandra Williams. 2012a. Measuring the Understandability of Deduction Rules for OWL. In *International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2012)*.

[NPPW12b] Tu Anh T. Nguyen, Richard Power, Paul Piwek, and Sandra Williams. 2012b. Planning Accessible Explanations for Entailments in OWL Ontologies. In *International Natural Language Generation Conference (INLG 2012)*, pages 110–114.

[Ngu11] Tu Anh T. Nguyen. 2011. Explaining Justifications in OWL DL Ontologies. In *Extended Semantic Web Conference (ESWC 2011) - PhD Symposium Poster*.

[NPPW10] Tu Anh T. Nguyen, Paul Piwek, Richard Power, and Sandra Williams. 2010. Justification Patterns for OWL DL Ontologies. Technical Report TR2011/05, The Open University, UK.

Chapter 2

Background

This chapter introduces key ideas and concepts that are used in the rest of this thesis. In addition, it provides an insight into the execution of commonly used reasoning algorithms for description logics, and algorithms for computing justifications of OWL ontologies. The purpose is to point out that those algorithms, although helpful and efficient, do not provide a basis for generating accessible explanations for entailments of OWL ontologies. Justifications and laconic justifications for entailments provide a good basis for generating such explanations; however, further investigations are required to help with understanding a justification, and to identify which of several justifications of an entailment provides the best basis for an explanation.

2.1 OWL Ontology and Description Logics

2.1.1 OWL Ontology

The term 'ontology' originally refers to a philosophical discipline which studies the nature and organisation of existence. It is then adopted in computer science as an *engineering artifact* that introduces vocabulary describing the relationships between concepts and objects in some domain of interest, and explicitly specifies the intended meaning of that vocabulary [Hor08]. Specifically, it provides a set of definitions for relevant concepts of

the domain, a set of objects as instances of the concepts, and a set of binary relationships between two objects, or an object and a value. The concepts, objects, and two types of binary relationships are referred to as *classes*, *individuals*, *object properties*, and *data properties*, respectively. For example, an ontology may consist of a class named 'Insectivore', described as a sub-class of 'Carnivore' with a distinguishing feature of eating only insects. It may also include an individual named 'Keroppi' which has four legs and is an instance of the class 'Insectivore'. In this example, 'eats' is an object property, and 'has leg number' is a data property.

An ontology is often encoded into a machine-processable form by using a formal logic-based language called an *ontology language*. The logical foundation provides a unique and unambiguous interpretation of the semantics of an ontology, for both humans and machines. Additionally, it allows *reasoning* with an ontology to infer implicit knowledge about classes, individuals, and properties in the form of logical statements called *entailments*. Each ontology language provides a set of *constructors* that combine classes, individuals, and properties into logical statements (called *axioms*) or expressions for specifying an ontology. A commonly used constructor is $\sqsubseteq$, which specifies the subsumption relationship between two classes, or even two complex class expressions. The more diverse the set of constructors, the more *expressive* the ontology language.

The latest standard ontology language is OWL (Web Ontology Language) [OWL12a], endorsed by the W3C (World Wide Web Consortium) in 2004. It provides a powerful set of constructors allowing one to build up highly expressive ontologies. A number of important ontologies have been built in OWL to standardize the representation of knowledge in different fields; examples are the gene ontology [ABB+00] in bio-informatics, the NCI (National Cancer Institute) Thesaurus [GFH+03], and the SNOMED ontology [SCC97] in health-care.

2.1.2 Description Logics

Description logics are *decidable fragments* of first-order predicate logic that are usually used in knowledge representation [BCM+03]. The decidability requirement here refers to a computational property of reasoning—that is, logical inferences in description logics are guaranteed to be computable within finite time. This, however, does not imply that

Table 2.1: OWL constructors for complex expressions, where 'a' and 'b' are individual names, 'C' and 'D' are class expressions, 'R_o' is an object property name, 'R_d' is a data property name, 'D_r' is a data range, 'D_t' is a data type, 'l' is a literal, and 'n' is a non-negative integer

ID	OWL Constructor	Semantics
1.	$C \sqcap D[\sqcap \ldots]$	The set of individuals that are instances of both C and D
2.	$C \sqcup D[\sqcup \ldots]$	The set of individuals that are instances C or D
3.	$\neg C$	The set of individuals that are not instances of C
4.	$\mathbf{One}(a, b[, \ldots])$	One of the individuals a, b
5.	$\exists R_o.C$	The set of individuals connected by R_o to an instance of C
6.	$\forall R_o.C$	The set of individuals connected by R_o to only instances of C
7.	$\exists R_o.\{a\}$	The set of individuals connected by R_o to individual a
8.	$\exists R_o.\mathbf{Self}$	The set of individuals connected by R_o to themselves
9.	$\geq nR_o.C$	The set of individuals connected by R_o to at least n instances of C
10.	$\leq nR_o.C$	The set of individuals connected by R_o to at most n instances of C
11.	$= nR_o.C$	The set of individuals connected by R_o to exactly n instances of C
12.	$\exists R_d.D_r$	The set of individuals connected by R_d to a D_r value
13.	$\forall R_d.D_r$	The set of individuals connected by R_d to only D_r values
14.	$\exists R_d.\{l\}$	The set of individuals connected by R_d to a literal l
15.	$\geq nR_d.D_r$	The set of individuals connected by R_d to at least n D_r values
16.	$\leq nR_d.D_r$	The set of individuals connected by R_d to at most n D_r values
17.	$= nR_d.D_r$	The set of individuals connected by R_d to exactly n D_r values
18.	$Inv(R_o)$	The inverse property of R_o

every inference can be computed within a reasonable time period. The time needed to compute an inference depends on the *computational complexity* of reasoning algorithms, which in turn depends on the expressiveness of the description logic. In fact, there is always a trade-off between the expressiveness of a description logic and the computational complexity of its reasoning algorithms.

Because of the decidability property, description logics are often used as the logical foundation of ontology languages. OWL [OWL04b], for instance, is underpinned by the description logic $\mathcal{SHOIN}$ [HPSvH03], and the latest OWL (i.e., OWL 2 [OWL12a]) is underpinned by the description logic $\mathcal{SROIQ}$ [HKS06], which is slightly more expressive than $\mathcal{SHOIN}$. In the next two sub-sections, the syntax and semantics of $\mathcal{SROIQ}$, and more importantly, how they are linked to constructors supported in OWL, are discussed.

2.1.3 Syntax

Description logics, including $\mathcal{SROIQ}$, have a vocabulary that contains the names of classes, individuals, and properties. The sets of class, individual, and property names are denoted by N_C, N_I, and N_R, respectively. Each name represents an *atomic* or *named*

entity. There are two additional class names, denoted by $\bot$ (Nothing) and $\top$ (Thing), that represent the empty class and the class of all individuals.

Description logics also support the construction of *anonymous* entities, often based on the construction of complex expressions. Constructors for complex expressions supported in OWL, including 17 constructors for class expressions and 1 for object property expressions, are listed in Table 2.1, with English glosses explaining their semantics. No anonymous data properties are supported in OWL.

Axioms provided by $\mathcal{SROIQ}$ are classified into three categories namely $RBox$, $TBox$, and $ABox$ [HKS06]. The $RBox$ (often denoted by $\mathcal{R}$) includes axioms that introduce a relationship between two properties such as $isBrotherOf \sqsubseteq isSiblingOf$ ("is brother of" is a sub-property of "is sibling of"). The $TBox$ (or $\mathcal{T}$) includes axioms that specify a relationship between two classes such as $Dolphin \sqsubseteq Mammal$ (Every dolphin is a mammal). The $ABox$ (or $\mathcal{A}$), on the other hand, includes axioms that assert the existence of individuals in the domain such as $Fish(nemo)$ (Nemo is a fish) and $isFriendOf(nemo, dory)$ (Nemo is a friend of Dory). Constructors for $RBox$, $TBox$, and $ABox$ axioms supported in OWL are listed in Table 2.2, with English sentences explaining their semantics.

2.1.4 Formal Semantics

As with first-order predicate logic, each description logic has a model-theoretic formal semantics, and is specified using the notion of *interpretations*. An interpretation of a description logic, often denoted by $\mathcal{I}$, can be understood as a potential 'world' created by a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, in which $\Delta^{\mathcal{I}}$ is the domain (i.e., the entirety of individuals), and $\cdot^{\mathcal{I}}$ is the interpretation function that maps the vocabulary to the domain [Rud11]—specifically, each individual name in N_I is mapped to an element in the domain, each class name in N_C to a sub-set of the domain, and each property name R in N_R to a (possibly empty) sub-set of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Given the interpretation of a named entity as above, the interpretations of other classes and class expressions in OWL can be extended in a natural way as follows (reproduced from [Hor11]):

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$

Table 2.2: OWL constructors for axioms, where 'a' and 'b' are individual names, 'A' is a class name, 'C' and 'D' are class expressions, 'R_o' and 'S_o' are object property names, 'R_d' and 'S_d' are data property names, 'D_r' is a data range, 'D_t' is a data type, and 'l' is a literal

ID	Type	OWL Constructor	Semantics
1.	RBox	$R_o \sqsubseteq S_o$	If an individual x is connected by R_o to an individual y then x is connected by S_o to y.
2.		$R_o \equiv S_o[\equiv \ldots]$	R_o and S_o are the same object property.
3.		$\mathbf{Dis}(R_o, S_o[, \ldots])$	There are no individual connected to an individual by both R_o and S_o.
4.		$\mathbf{Invs}(R_o, S_o)$	If an individual x is connected by R_o to an individual y then y is connected by S_o to x, and vice versa.
5.		$\mathbf{Fun}(R_o)$	Each individual is connected by R_o to at most one individual.
6.		$\mathbf{InvFun}(R_o)$	For each individual x, there is at most one individual y connected by R_o to x.
7.		$\mathbf{Ref}(R_o)$	Each individual is connected by R_o to itself.
8.		$\mathbf{Irr}(R_o)$	No individual is connected by R_o to itself.
9.		$\mathbf{Sym}(R_o)$	If an individual x is connected by R_o to an individual y then y is connected by R_o to x.
10.		$\mathbf{Asym}(R_o)$	If an individual x is connected by R_o to an individual y then y cannot be connected by R_o to x.
11.		$\mathbf{Tra}(R_o)$	If an individual x is connected by R_o to an individual y that is connected by R_o to an individual z, then x is also connected by R_o to z.
12.		$R_d \sqsubseteq S_d$	If an individual x is connected by R_d to a literal l then x is connected by S_d to l.
13.		$R_d \equiv S_d[\equiv \ldots]$	R_d and S_d are the same data property.
14.		$\mathbf{Dis}(R_d, S_d[, \ldots])$	There are no individual connected to a literal by both R_d and S_d.
15.		$\mathbf{Fun}(R_d)$	Each individual is connected by R_d to at most one literal.
16.	TBox	$C \sqsubseteq D$	All instances of C are instances of D.
17.		$C \equiv D[\equiv \ldots]$	All instances of C are instances of D, and vice versa.
18.		$\mathbf{Dis}(C, D[, \ldots])$	No instance of C is an instance of D.
19.		$\mathbf{DisUni}(A, C, D[, \ldots])$	Each instance of A is an instance of either C or D but not both, and vice versa.
20.		$\mathbf{Dom}(R_o, C)$	Only instances of C are connected by R_o to an individual.
21.		$\mathbf{Rng}(R_o, C)$	Each individual can only be connected by R_o to instances of C.
22.		$\mathbf{Dom}(R_d, C)$	Only instances of C are connected by R_d to a literal.
23.		$\mathbf{Rng}(R_d, D_r)$	Each individual can only be connected by R_d to D_r values.
24.	ABox	$C(a)$	Individual a is an instance of C.
25.		$R_o(a, b)$	Individual a is connected by R_o to individual b.
26.		$\neg R_o(a, b)$	Individual a cannot be connected by R_o to individual b.
27.		$\mathbf{Sam}(a, b[, \ldots])$	a and b are the same individual.
28.		$\mathbf{Dif}(a, b[, \ldots])$	a and b are different individuals.
29.		$R_d(a, l \star D_t)$	Individual a is connected by R_d to a literal l of type D_t.
30.		$\neg R_d(a, l \star D_t)$	Individual a cannot be connected by R_d to a D_t value of l.

- $\bot^{\mathcal{I}} = \emptyset$
- $\{a, b\}^{\mathcal{I}} = \{a^{\mathcal{I}}, b^{\mathcal{I}}\}$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\exists R_o.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}}.(a, b) \in R_o^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
- $(\forall R_o.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b \in \Delta^{\mathcal{I}}.(a, b) \in R_o^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
- $(\exists R_o.\mathbf{Self})^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (a, a) \in R_o^{\mathcal{I}}\}$
- $(\geq n R_o)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \,\middle|\, |\{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R_o^{\mathcal{I}}\}| \geq n\right\}$
- $(\leq n R_o)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \,\middle|\, |\{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R_o^{\mathcal{I}}\}| \leq n\right\}$

The interpretations of OWL axioms (denoted by $\mathcal{I} \models \alpha$ where α is an arbitrary axiom) can also be extended as follows (reproduced from [Hor11]):

- $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models C \equiv D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $D^{\mathcal{I}} \subseteq I^{\mathcal{I}}$
- $\mathcal{I} \models \mathbf{Dis}(R_o, S_o)$ if $R_o^{\mathcal{I}} \cap S_o^{\mathcal{I}} = \emptyset$
- $\mathcal{I} \models \mathbf{Sym}(R_o)$ if $\forall a, b, (a, b) \in R_o^{\mathcal{I}}$ implies $(b, a) \in R_o^{\mathcal{I}}$
- $\mathcal{I} \models \mathbf{Tra}(R_o)$ if $\forall a, b, c, (a, b) \in R_o^{\mathcal{I}}$ and $(b, c) \in R_o^{\mathcal{I}}$ implies $(a, c) \in R_o^{\mathcal{I}}$
- $\mathcal{I} \models C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I} \models R_o(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R_o^{\mathcal{I}}$

2.2 Reasoning in Description Logics

2.2.1 Reasoning Tasks

As mentioned before, it is possible to reason with an ontology to produce entailments. In theory, there should be no limitation on reasoning tasks that can be performed; in practice,

description logics support only certain standard reasoning tasks. Regarding inferences for classes, they support the following four tasks:

Satisfiability checking Check if the definition of a class is satisfiable (i.e., can contain one or more individuals), or unsatisfiable (i.e., cannot contain any individual)

Subsumption checking Check if one class is more general than (i.e., subsumes) another class

Equivalence checking Check if the definitions of two classes are semantically equivalent

Disjointness checking Check if two classes are disjoint with each other

Among these tasks, satisfiability checking is the key because other tasks can be reduced to it. Assume that C and D are two class expressions, $C \sqsubseteq D$ if and only if $(C \sqcap \neg D)$ is unsatisfiable. Similarly, $C \equiv D$ if and only if both $(C \sqcap \neg D)$ and $(\neg C \sqcap D)$ are unsatisfiable. For disjointness checking, C and D are disjoint if and only if $(C \sqcap D)$ is unsatisfiable.

For inferences with $ABox$ assertions, description logics support two additional reasoning tasks as follows:

Instance checking Check if an individual can be entailed as an instance of a class

Consistency checking Check if the assertions in $ABox$ are consistent with respect to all the definitions in $RBox$ and $TBox$

Instance checking can be reduced to consistency checking: assume that a is an individual name then $C(a)$ is entailed from an ontology $\mathcal{O}$ if and only if $(\mathcal{O} \cup \{\neg C(a)\})$ is inconsistent. Satisfiability checking can also be reduced to consistency checking: the class expression C is satisfiable if and only if $\{C(a)\}$ is consistent, where a is an arbitrary individual name. This means that all standard reasoning tasks are reducible to consistency checking, so an efficient consistency checking algorithm is sufficient to ensure the efficiency of reasoning in a description logic. It should be noted that to be able to reduce all standard reasoning tasks to satisfiability as well as consistency checking, the description logic needs to support both *conjunction* and *negation* of any arbitrary class expression.

To perform the above-mentioned reasoning tasks in description logics, three main kinds of reasoning algorithms have been proposed and used so far, namely *structural subsumption*

algorithm, *tableau-based* algorithm, and *resolution-based* algorithm. These algorithms are described in detail in the next three sub-sections.

2.2.2 Structural Subsumption Algorithms

Structural subsumption algorithms [BS01] are designed to check for subsumption between two class expressions. They are often used in inexpressive description logics in which subsumption checking is not reducible to satisfiability or consistency checking. Their execution consists of two phases: *normalisation* and *comparison*. In the first phase, the class expressions are transformed into *normal forms*; in the second phase, the syntactic structure of the normal forms are recursively compared to check for subsumption. Let's examine how such an algorithm [BCM+03] determines subsumption between two class expressions in $\mathcal{FL}-$ [BL84], an inexpressive description logic that does not support the negation or disjunction operator. In the normalisation phase, both class expressions are converted into the following normal forms (reproduced from [BCM+03]):

$A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_{o1}.C_1 \sqcap \ldots \sqcap \forall R_{on}.C_n$, and

$B_1 \sqcap \ldots \sqcap B_k \sqcap \forall S_{o1}.D_1 \sqcap \ldots \sqcap \forall S_{ol}.D_l$

where $A_1, \ldots, A_m$ and $B_1, \ldots, B_k$ are class names, $R_{o1}, \ldots, R_{on}$, $S_{o1}, \ldots, S_{ol}$ are object property names, $C_1, \ldots, C_n$ and $D_1, \ldots, D_n$ are arbitrary class expressions. In the comparison phase, the first normal form is subsumed by the second one, and so the first class expression is subsumed by the second one, if and only if the following two conditions hold [BCM+03]:

1. for all i, $1 \leq i \leq k$, there exists j, $1 \leq j \leq m$ such that $B_i = A_j$
2. for all i, $1 \leq i \leq l$, there exists j, $1 \leq j \leq n$ such that $S_{oi} = R_{oj}$ and $C_j \sqsubseteq D_i$

It is not difficult to see that it is nearly impossible to extend this algorithm for more expressive description logics such as $\mathcal{ALC}$ [SS91], which fully supports negation, disjunction, and existential restriction (i.e., $\exists R.C$). For such description logics, tableau-based and resolution-based algorithms are used to perform reasoning.

Table 2.3: Transformation rules for consistency checking in tableau-based algorithms (reproduced from [Hor11])

Rule	Condition and Action
$\sqcap -$ **rule**	If $(C \sqcap D)(x) \in \mathcal{A}$, but $C(x) \notin \mathcal{A}$ and $D(x) \notin \mathcal{A}$ then create $\mathcal{A}' = \mathcal{A} \cup \{C(x), D(x)\}$.
$\sqcup -$ **rule**	If $(C \sqcup D)(x) \in \mathcal{A}$, but $C(x) \notin \mathcal{A}$ and $D(x) \notin \mathcal{A}$ then create $\mathcal{A}' = \mathcal{A} \cup \{C(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{D(x)\}$.
$\exists -$ **rule**	If $(\exists R.C)(x) \in \mathcal{A}$, but $R(x, y) \notin \mathcal{A}$ and $C(y) \notin \mathcal{A}$ for some y then create $\mathcal{A}' = \mathcal{A} \cup \{C(y), R(x, y)\}$.
$\forall -$ **rule**	If $(\forall R.C)(x) \in \mathcal{A}$, and $R(x, y) \in \mathcal{A}$, but $C(y) \notin \mathcal{A}$ then create $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$.

2.2.3 Tableau-Based Algorithms

Tableau-based algorithms [BS01] are refutation-based algorithms designed to perform reasoning in expressive description logics. They have been employed in many automated reasoners for OWL ontologies; examples are FaCT++ [TH06], HermiT [MSH07], Pellet [SPG+07], RACER [HM01], and fuzzyDL [fuz]. The first tableau-based algorithm was introduced by Schmidt-Schauß and Smolka [SS91] for checking satisfiability of a class expression in $\mathcal{ALC}$. It was then extended to cover more expressive description logics [BS99, HS99], and perform consistency checking [HM00]. Since other reasoning tasks are reducible to consistency checking, they are also supported by tableau-based algorithms.

The main idea behind tableau-based algorithms is illustrated here through an example taken from [BCM+03]—that is, examining whether $(\exists R_o.A) \sqcap (\exists R_o.B)$ is subsumed by $\exists R_o.(A \sqcap B)$. Firstly, the subsumption problem is reduced to an equivalent satisfiability problem—that is, checking whether the class expression $C = (\exists R_o.A) \sqcap (\exists R_o.B) \sqcap \neg(\exists R_o.(A \sqcap B))$ is unsatisfiable. This is done by applying double negation elimination (i.e., $\neg\neg C$ is equivalent to C), and De Morgan's rules (i.e., $\neg(C \sqcap D) = (\neg C \sqcup \neg D)$, and $\neg(C \sqcup D) = (\neg C \sqcap \neg D)$). Next, the class expression is transformed to an equivalent *negation normal form* in which negations only occur immediately in front of class names. The negation normal form of C in this example is $C_0 = (\exists R_o.A) \sqcap (\exists R_o.B) \sqcap \forall R_o.(\neg A \sqcup \neg B)$. Now the algorithms check whether C_0 is unsatisfiable.

In order to check for the satisfiability of C_0, the algorithm tries to check whether the $ABox$ $\mathcal{A}_0 = \{C_0(a)\}$ is consistent, where a is an arbitrary individual name. The algorithm starts with $\mathcal{A}_0$, and applies *transformation rules* shown in Table 2.3. After applying a rule, one

Table 2.4: Applications of tableau-based transformation rules in the example

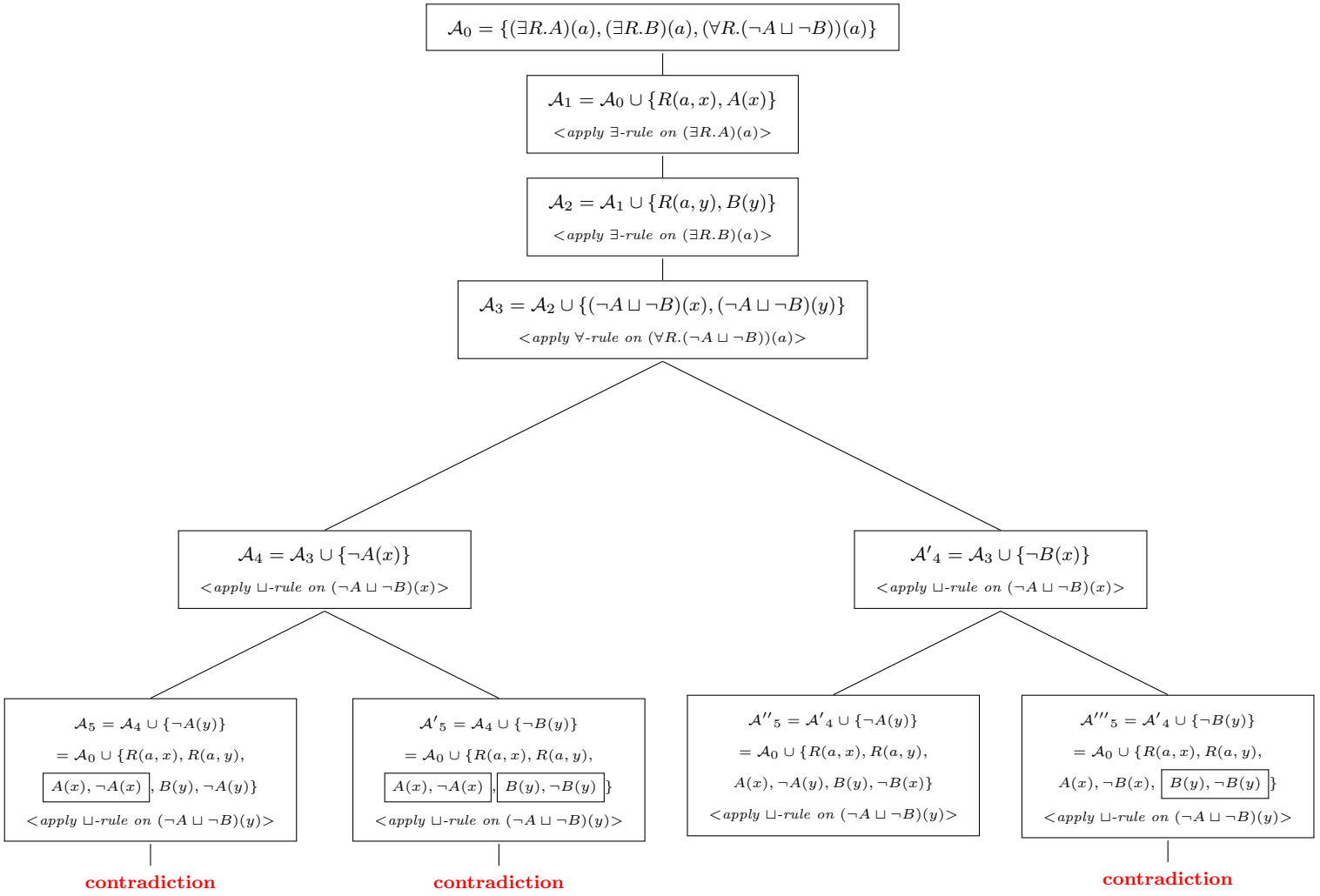


Table 2.5: Translation ($\rightsquigarrow$) of description logic axioms into first-order clauses (reproduced from [Mot09]); underlined literals are those the algorithm selects to be unified in the next steps

$$
\begin{array}{rcll}
\top \sqsubseteq Q_1 \sqcup Q_2 & \rightsquigarrow & Q_1(x) \vee \underline{Q_2(x)} & (1) \\
Q_1 \sqsubseteq \forall S.\neg A & \rightsquigarrow & \neg Q_1(x) \vee \underline{\neg S(x,y)} \vee \neg A(y) & (2) \\
Q_2 \sqsubseteq \exists R.B & \rightsquigarrow & \neg Q_2(x) \vee \underline{R(x, f(x))} & (3) \\
Q_2 \sqsubseteq \exists R.B & \rightsquigarrow & \neg Q_2(x) \vee \underline{B(f(x))} & (4) \\
B \sqsubseteq C & \rightsquigarrow & \neg B(x) \vee \underline{C(x)} & (5) \\
\exists R.C \sqsubseteq D & \rightsquigarrow & D(x) \vee \underline{\neg R(x,y)} \vee \neg C(y) & (6) \\
S(a,b) & \rightsquigarrow & \underline{S(a,b)} & (7) \\
A(b) & \rightsquigarrow & \underline{A(b)} & (8) \\
\neg D(a) & \rightsquigarrow & \underline{\neg D(a)} & (9)
\end{array}
$$

or more new *ABox*es will be created, and the algorithm will recursively apply the rules on the new *ABox*es. The algorithm stops when either there are no rules that can be applied, or obvious contradictions are found in the *ABox*es. The resulting *ABox*es are called *complete ABox*es. Among the set of complete *ABox*es, if there exists at least one that does *not* contain any obvious contradiction, then the algorithm concludes that $\mathcal{A}_0$ is consistent, and so C_0 is satisfiable. Otherwise, $\mathcal{A}_0$ is inconsistent, and C_0 is unsatisfiable.

Table 2.4 shows all the applications of transformation rules on $\mathcal{A}_0$ in our example. The algorithm stops with four complete *ABox*es, one of which is consistent. Therefore, the algorithm concludes that C_0 is satisfiable, and so $(\exists R_o.A) \sqcap (\exists R_o.B)$ is not subsumed by $\exists R_o.(A \sqcap B)$. Among the transformation rules in Table 2.3, those for disjunction and at-most restriction are *non-deterministic* because they may result in multiple new *ABox*es from a given *ABox* $\mathcal{A}$. In such cases, as shown in the example, $\mathcal{A}$ is consistent if and only if at least one of the new *ABox*es is consistent.

2.2.4 Resolution-Based Algorithms

Resolution [BG01] is a refutation-based theorem proving method for checking the satisfiability of a set of first-order clauses. It systematically and exhaustively applies a set of inference rules to the clauses in order to search for a contradiction. The output of this searching is a resolution proof showing step by step the application of the rules. If a contradiction is found, then the input clause set is unsatisfiable; otherwise, it is satisfiable.

Resolution-based algorithms use resolution to check for satisfiability of a concept or consistency of an *ABox* in expressive description logics. Since resolution is only applicable

Table 2.6: Resolution of first-order clauses in Table 2.5 (reproduced from [Mot09]); each R(i + j) is a resolution of two clauses at the indexes (i) and (j), and underlined literals are those the algorithm selects to be unified in the next steps

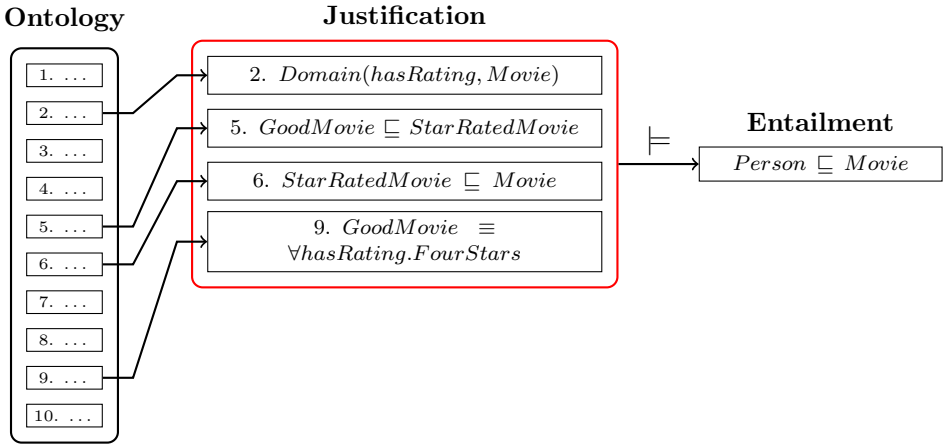
$D(x) \vee \neg Q_2(x) \vee \underline{\neg C(f(x))}$	R(3 + 6)	(10)
$D(x) \vee \neg Q_2(x) \vee \underline{\neg B(f(x))}$	R(10 + 5)	(11)
$D(x) \vee \underline{\neg Q_2(x)}$	R(11 + 4)	(12)
$D(x) \vee \underline{Q_1(x)}$	R(12 + 1)	(13)
$\underline{\neg Q_1(a)} \vee \neg A(b)$	R(2 + 7)	(14)
$\underline{D(a)} \vee \neg A(b)$	R(13 + 14)	(15)
$\underline{\neg A(b)}$	R(9 + 15)	(16)
$\square$	R(8 + 16)	(17)

to first-order clauses, these algorithms usually consist of two phases: *translation* and *resolution*. First, all description logic axioms are translated into semantically equivalent first-order clauses, then resolution algorithms are used to perform satisfiability or consistency checking. One important advantage of these algorithms is that standard resolution provers, such as Bliksem [Bli], SPASS [WDF+09], and Otter [Ott], can be re-used. However, the translation phase adds extra overhead to their efficiency. Resolution-based algorithms have been used in several automated reasoners for OWL ontologies such as KAON2 [KAO13, HMS04], and MSPASS [MSP].

The underlying idea of resolution-based algorithms is illustrated here through an example taken from [Mot09]—that is, examining whether an ontology $\mathcal{O} = \{\exists S.A \sqsubseteq \exists R.B, B \sqsubseteq C, \exists R.C \sqsubseteq D, S(a,b), A(b)\}$ entails $D(a)$. Recall that $\mathcal{O} \models D(a)$ if and only if $(\mathcal{O} \cup \{\neg D(a)\})$ is inconsistent. In the translation phase, the first axiom is structurally transformed into $\top \sqsubseteq (\forall S.\neg A) \sqcup (\exists R.B)$ then into a set of axioms $\{\top \sqsubseteq Q_1 \sqcup Q_2, Q_1 \sqsubseteq \forall S.\neg A, Q_2 \sqsubseteq \exists R.B)\}$; thereafter, the whole ontology is translated into first-order clauses as shown in Table 2.5.

In the resolution phase, the first-order clauses are resolved to obtain new clauses. The resolutions are recursively applied on the clauses until either a contradiction is found or no further resolutions can be applied. All the resolutions in our example are shown in Table 2.6. The algorithm derives an empty clause, suggesting that there is a contradiction within the set $\mathcal{O} \cup \{\neg D(a)\}$. In other words, the ontology entails $D(a)$.

Figure 2.1: A justification for an entailment of an ontology



2.3 Justifications for Entailments

2.3.1 Justifications as Explanations

A *justification* for an entailment of an ontology is defined as any minimal subset of the ontology from which the entailment can be drawn [Kal06]. The minimality requirement here means that if any axiom is removed from the justification, the entailment will no longer be inferable. Figure 2.1 shows how a justification for an entailment is linked to axioms in the ontology. It should be noted that there may be multiple justifications for a given entailment from an ontology.

Justifications are useful for understanding as well as diagnosing and repairing an undesirable entailment [Kal06]. Such an entailment signals that something has gone wrong, and that the ontology needs to be debugged. Before starting to repair the ontology, it is necessary to find out the reasons why that entailment was drawn in order to identify which axiom or axioms need to be corrected. Without the justification support, a manual examination of the entire ontology is required. In the case of very large ontologies, such as SNOMED [SCC97] which contains over 500,000 axioms, such an examination is very difficult and time consuming.

Our corpus-based study on justifications for entailments from real world OWL ontologies (reported in Chapter 4) shows that the average ontology size (i.e., the average number of axioms) of our corpus is 1,131 whereas the average justification size for subsumption entailments is 8. This study also shows variations in justification size from 1 up to 192

axioms, with 69.4% of justifications (or 153,808 justifications) consisting of ten or fewer axioms. Additionally, the multiplicity property of justifications for an entailment enables the exploration of different reasons for the entailment to follow.

2.3.2 Computing Justifications

Given the above-mentioned benefits of justifications, a number of algorithms have been proposed to compute justifications for an entailment [BH95, SC03, Kal06, BPnS07, SQJH08, JQH09, Hor11]. These algorithms are known technically as *axiom pinpointing* algorithms. This sub-section provides an insight into these algorithms, and shows that even though they can determine the sets of premises for an entailment efficiently, their implementation does not provide useful information for explaining it.

Generally, existing axiom pinpointing algorithms are classified into two main categories, namely *black-box* (or reasoner-independent), and *glass-box* (or reasoner-dependent) [PSK05]. A black-box algorithm uses an automated reasoner as an external subroutine which provides answers for queries of whether an entailment follows from a set of axioms. These answers are taken as inputs for computing justifications for the entailment. A glass-box algorithm, on the other hand, modifies the internal reasoning algorithm of a reasoner to enable the computation of justifications. Compared with glass-box algorithms, black-box algorithms are simpler to implement as they rely on available reasoners [Kal06], but they usually suffer higher computational complexity due to a number of calls to a reasoner [Stu08].

Recall that there may be multiple justifications for an entailment of an ontology. For computing *all* possible justifications for an entailment, the approach in earlier work is to first develop an algorithm for computing a single justification, then develop another algorithm that uses the former one as a sub-routine to compute all remaining justifications. Since each algorithm can be either glass-box or black-box, an algorithm for computing *all* justifications of an entailment often falls into one of the three possible categories [Hor11]: *pure glass-box* algorithms, *pure black-box* algorithms, and *hybrid black-box glass-box* algorithms.

Pure Glass-Box Algorithms

A pure glass-box algorithm [Hor11] is a glass-box algorithm that computes all justifications of an entailment by using only glass-box sub-routines for computing individual justifications. Existing algorithms of this type include those of Schlobach et al. [SC03, SHCH07],

Meyer et al. [MLBP06], and Lam et al. [LSPV08] for the description logic $\mathcal{ALC}$, those of Baader et al. [BPnS07, BS08] for the description logic $\mathcal{EL}+$, and that of Kalyanpur [Kal06] for the description logic $\mathcal{SHOIN}$.

These algorithms follow Baader and Hollunder's idea of 'tracing' and 'labelling' the execution of a tableau-based algorithm [BH95]. According to this idea, by tracing the execution of a tableau-based algorithm, and labelling each literal asserted into an *ABox* after applying a transformation rule (as explained in Section 2.2.3) with information of how it is generated, the labels of assertions in a contradictory complete *ABox* can be used to compute a single justification. By checking all contradictory complete *ABox*es, all justifications of the entailment can be computed.

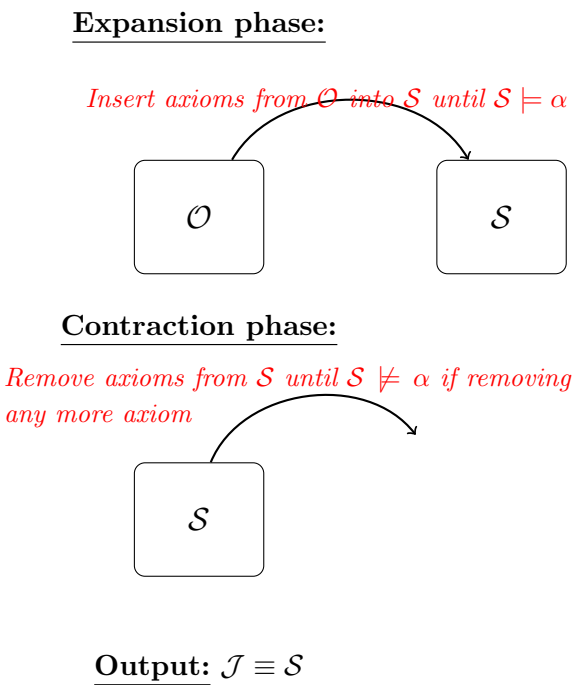
Pure Black-Box Algorithms

A pure black-box algorithm [Hor11] is a black-box algorithm that computes all justifications of an entailment by using only back-box sub-routines for computing individual justifications. As opposed to pure glass-box algorithms, pure black-box algorithms require two separate algorithms, one for computing a single justification, and the one for computing all justifications.

In a basic algorithm to compute a justification $\mathcal{J}$ for an entailment α from an ontology $\mathcal{O}$ [Hor11], the computation starts with a sub-set $\mathcal{S}$ of $\mathcal{O}$. If $\mathcal{S}$ does not entail α (i.e., $\mathcal{S} \not\models \alpha$), the algorithm keeps moving new axioms from $\mathcal{O}$ to $\mathcal{S}$ until $\mathcal{S}$ entails α (this phase is called *expansion*); otherwise, the expansion phase is skipped. The result of this phase is a set of axioms $\mathcal{S}$ that is a super-set of $\mathcal{J}$. Thereafter, axioms are gradually removed from $\mathcal{S}$ until $\mathcal{S}$ does not entail α if removing any further axiom (this phase is called *contraction*). The set of axioms remained in $\mathcal{S}$ is a justification for α (i.e., $\mathcal{J} \equiv \mathcal{S}$). The two phases of this algorithm are depicted in Figure 2.2.

To find all justifications for an entailment from an ontology $\mathcal{O}$, Kalyanpur et al. [KPHS07] proposed an algorithm that is based on Reiter's Hitting Set Tree algorithm [Rei87]. In this algorithm, a random justification $\mathcal{J}$ is first computed by using the above-mentioned algorithm, then each axiom α in $\mathcal{J}$ is removed individually from $\mathcal{O}$ resulting in a new ontology $\mathcal{O}' = \mathcal{O} \setminus \{\alpha\}$. The algorithm then computes a new justification $\mathcal{J}'$ for the entailment from $\mathcal{O}'$; in other words, it searches for a new justification $\mathcal{J}'$ from $\mathcal{O}$ that does *not* include the axiom α. This process is systematically iterated until all remaining

Figure 2.2: The basic algorithm for computing a single justification $\mathcal{J}$ for an entailment α from an ontology $\mathcal{O}$; initially, $\mathcal{S} \subseteq \mathcal{O}$



justifications are found.

Hybrid Black-Box Glass-Box Algorithms

In a hybrid black-box glass-box algorithm [Hor11], individual justifications are computed with a pure glass-box algorithm, and all justifications with a pure black-box one. Existing hybrid black-box glass-box algorithms include those of Kalyanpur [Kal06, KPHS07], and Suntisrivaraporn [Sun09].

2.3.3 Laconic justifications

Axioms in an ontology reflect developers' intuitions about a domain, and are usually manually asserted. They can be very long, and conjoin several complex expressions. Since axioms in a justification are directly taken from asserted axioms in the ontology, some of them may contain parts that are *unnecessary* for the entailment to follow [HPS08b, Hor11]. The presentation of such parts in an justification may cause distractions for readers when they try to understand the inference. Additionally, when repairing an undesirable entailment, people tend to remove one or more entire axioms that they believe to be incorrect. If these axioms contain unnecessary parts, removing them would cause the ontology to be *over-repaired* (i.e., removing more information than necessary) [HPS08b,

Table 2.7: Examples of laconic justifications

Type	Example Justification	Laconic Justification
$\bot$-*superfluity*	**Entailment:** $Person \sqsubseteq Movie$ **Justification:** 1. $GoodMovie \equiv \forall hasRating.FourStars$ 2. $Domain(hasRating, Movie)$ 3. $GoodMovie \sqsubseteq StarRatedMovie$ 4. $StarRatedMovie \sqsubseteq Movie$	**Entailment:** $Person \sqsubseteq Movie$ **Justification:** 1'. $\forall hasRating.\bot \sqsubseteq GoodMovie$ 2. $Domain(hasRating, Movie)$ 3. $GoodMovie \sqsubseteq StarRatedMovie$ 4. $StarRatedMovie \sqsubseteq Movie$
$\top$-*superfluity*	**Entailment:** $A \sqsubseteq B$ **Justification:** 1. $A \sqsubseteq C \sqcap D$ 2. $C \sqsubseteq B$	**Entailment:** $A \sqsubseteq B$ **Justification:** 1'. $A \sqsubseteq C$ 2. $C \sqsubseteq B$
Weakening	**Entailment:** $A \sqsubseteq B$ **Justification:** 1. $A \sqsubseteq \geq 4R_o.C$ 2. $\exists R_o.C \sqsubseteq B$	**Entailment:** $A \sqsubseteq B$ **Justification:** 1'. $A \sqsubseteq \geq 1R_o.C$ 2. $\exists R_o.C \sqsubseteq B$

Hor11].

To solve this problem, Horridge et al. have defined a new class of fine-grained justifications called *laconic justifications*. A laconic justification [HPS08b, Hor11] is a justification in which no axioms contain any superfluous parts. The first example in Table 2.7 shows a laconic justification of the justification in Table 1.1. In this example, the first axiom in the original justification, which says "A good movie is anything that has only four stars as ratings" is transformed to a new axiom, which says "Everything that has no rating at all is a good movie", because only this piece of information is required for the entailment. This transformation is in fact caused by the replacement of the superfluous class name '$FourStarRating$' by '$\bot$'—this replacement is named technically $\bot$-*superfluity* [Hor11]. Other axioms in the original justification are unchanged because they contain no superfluous parts.

In addition to $\bot$-*superfluity*, Horridge et al. [Hor11] also support two additional types of replacement called $\top$-*superfluity* and *weakening*. In $\top$-*superfluity*, the superfluous class name is replaced by the class name '$\top$'—e.g., the replacement of 'D' by '$\top$' in the second example in Table 2.7. In *weakening*, the cardinality restrictions are weakened by either increasing or decreasing the cardinality values—e.g., the decrease of the cardinality value '4' to '1' in the last example in Table 2.7.

2.4 Discussion and Conclusions

To generate an explanation for an entailment computed by an automated reasoner, an ad-hoc solution is to adjust the *implementation* of the internal reasoning algorithm by inserting print-statements or sophisticated codes at appropriate places so that the outputs can be used to form an explanation [MB95]. This approach is, however, as inappropriate for explanation purposes, even for inexpressive algorithms such as structural subsumption [MB95].

Although explanations should be independent of the implementation of a reasoning algorithm, there exists a thought that they have to be tightly bound to the main steps in the execution of the reasoning algorithm in order to be useful for end-users [McG96, BFH+99]. However, it can be seen from the descriptions of three kinds of reasoning algorithms that their underlying ideas are not intuitive for human understanding. Among these three kinds, structural subsumption algorithms are the most intuitive; however, they are limited to only some inexpressive description logics.

Tableau-based and resolution-based algorithms existing for OWL are much less intuitive. In tableau-based algorithms, unintuitive transformations such as De Morgan's rules are applied to reduce a subsumption checking problem to an equivalent consistency checking problem. Such transformations may cause the loss of reference to the original problem. Additionally, transformation rules used in the later stage, such as those listed in Table 2.3, are refutation-based and not for human understanding. In resolution-based algorithms, the translation of description logics axioms to first-order clauses, and the resolution of first-order clauses make them even less intuitive than tableau-based algorithms.

The above analyses indicate that instead of providing explanations that are tightly bound to either the implementation or the execution of a reasoning algorithm, a better approach is to build up a *reasoner-independent* explanation system or, more generally, a *reconstructive* explanation system—a stand-alone system in which additional knowledge and methods can be added to produce reasoner-independent and user-oriented explanations—as suggested by Wick and Thompson for building an explanation facility in expert systems [WT92].

Similar to reasoning algorithms, algorithms for computing justifications are refutation-based, and do not provide useful information for generating reasoner-independent and user-

oriented explanations. Their outputs, justifications and laconic justifications, however, provide a good basis for such explanations. A justification provides a set of premises for an entailment, and is found to be helpful for understanding as well as debugging undesirable entailments [Kal06]. A laconic justification provides a more precise set of premises by eliminating all unnecessary parts in axioms. However, as discussed before, understanding how to get from a set of premises to a conclusion is not always easy. Additionally, Horridge et al.'s study [HPS09b, Hor11] shows that the transformation from an axiom in the original justification to one in the laconic justification can be very difficult to understand, even for OWL experts. Therefore, investigations how to generate explanations from justifications and/or laconic justifications are necessary.

In conclusion, there is a need for a reconstructive explanation system that can generate reasoner-independent and user-oriented explanations for entailments of OWL ontologies. Justifications and laconic justifications (but not the computation of them) provide a good basis for such explanations, but investigations how to generate such explanations are necessary. Moreover, it is unknown which of several justifications and laconic justifications for an entailment provide the best basis for explaining the entailment.

Chapter 3

Related Work

This chapter describes prior work in explaining description logic-based reasoning and mathematical theorems. Most of these work follow the *reconstructive* paradigm in generating explanations for reasoning (discussed in Section 2.4). The purpose of this chapter is to point out that the introduction of lemmas representing steps in inferring an entailment from a justification can help with understanding the entailment, and a number of ideas and methods suggested by earlier work on explaining mathematical theorems can help to improve the quality of explanations. However, investigations are required to identify which inference patterns are suitable for single inference steps in an explanation, as well as which possible lemmas provides the best subdivision of an inference into multiple steps.

3.1 Explaining Reasoning in Description Logics

3.1.1 Reasoning-Based Explanations

The first notable explanation facility for description logic-based reasoning was developed by McGuinness [McG96] in CLASSIC [BBMR89]—a knowledge representation system that is underpinned by an inexpressive description logic (described in [BMPS$^+$91]), and is equipped with a structural subsumption algorithm (described in Section 2.2.2). In this early work, explanations for reasoning are thought to have to be tightly correlated with the

Table 3.1: Nine examples of deductions rules used in McGuinness's system (reproduced from [McG96]), with '**Prim**' means primitives, '**Ref**' means reflexivity, '**Trans**' means transitivity, '**AtLst**' means at least, 'R_o' is an object property name, 'C', 'D', and 'E' are class expressions, $\tilde{\alpha}$ and $\tilde{\theta}$ are sets of class names. According to the **Prim** rule, a class expression is subsumed by another class expression if the set of class names in the first expression is a superset of that of the second one—for instance, $Lady \sqcap Young \sqsubset Lady$ because $\{Lady\} \subset \{Lady, Young\}$.

Type	**Rule Name**	**Deduction Rule**
Normalisation	**AllNothing**	$\forall R_o.\bot \equiv \leq 0R_o$
	AtLst0	$\geq 0R_o.\top \equiv \top$
	AllThing	$\forall R_o.\top \equiv \top$
Comparison	**AtLst**	$n > m$ $\rightarrow (\geq nR_o) \sqsubseteq (\geq mR_o)$
	All	$C \sqsubseteq D$ $\rightarrow \forall R_o.C \sqsubseteq \forall R_o.D$
	Prim	$\tilde{\alpha} \subseteq \tilde{\theta}$ $\rightarrow$ (**prim** $\tilde{\theta}$) $\sqsubset$ (**prim** $\tilde{\alpha}$)
Description Logics	**Trans**	$C \sqsubseteq D$, $D \sqsubseteq E$ $\rightarrow C \sqsubseteq E$
	Ref	$C \sqsubseteq C$
	Thing	$C \sqsubseteq \top$

execution of the reasoning algorithm [McG96]. They are provided in the form of *proofs*, or *fragments of proofs*, that explain in a declarative way how the reasoning algorithm *proves* an inference.

McGuinness focuses only on proofs for subsumption inferences. These proofs are generated from a set of inference rules [Bor92] that correspond to steps in either the normalisation or comparison phases of the structural subsumption algorithm. This rule set also includes rules corresponding to fundamental inferences in description logics. Table 3.1 shows nine inference rules, three for each type, used in McGuinness's system. The **Prim** (i.e., primitives) rule is for the comparison phase, and refers to subsumption between two class expressions. Recall that in the normalisation phase, class expressions are normalised into the form of conjunctions of class names. According to the **Prim** rule, a class expression is subsumed by another if the set of class names in the first expression is a superset of that of the second one. For instance, $Lady \sqcap Young \sqsubset Lady$ because $\{Lady\} \subset \{Lady, Young\}$.

It can be seen that many inference rules in Table 3.1 are trivial, such as the normalisation rules, **Prim** (i.e., primitives), **Ref** (i.e., reflexivity), and **Thing**. This may result

Table 3.2: An example of a proof generated by McGuinness's explanation system (reproduced from [McG96]) with each line justified by the name of the deduction rule used to deduce it and the line numbers of the associated premises

1.	$\geq 3hasGrape \sqsubseteq \geq 2hasGrape$	**AtLst**
2.	$(\geq 3hasGrape) \sqcap (\mathbf{prim}GoodWine) \sqsubseteq \geq 2hasGrape$	**AndL**, 1
3.	$(\mathbf{prim}GoodWine) \sqsubseteq (\mathbf{prim}Wine)$	**Prim**
4.	$(\geq 3hasGrape) \sqcap (\mathbf{prim}GoodWine) \sqsubseteq (\mathbf{prim}Wine)$	**AndL**, 3
5.	$A \equiv (\geq 3hasGrape) \sqcap (\mathbf{prim}GoodWine)$	**Told**
6.	$A \sqsubseteq (\mathbf{prim}Wine)$	**Eq**, 4, 5
7.	$Wine \equiv \top \sqcap (\mathbf{prim}Wine)$	**AndEq**
8.	$A \sqsubseteq (\sqcap(\mathbf{prim}Wine))$	**Eq**, 7, 6
9.	$A \sqsubseteq \geq 2hasGrape$	**Eq**, 5, 2
10.	$A \sqsubseteq (\geq 2hasGrape) \sqcap (\mathbf{prim}Wine)$	**AndR**, 9, 8

in long proofs with many trivial steps. In such proofs, the main line of reasoning is usually not visible to the readers. As an example, Table 3.2 shows a proof constructed by McGuinness's system to explain why $(\geq 3hasGrape) \sqcap (\mathbf{prim}\ Good\ Wine) \sqsubseteq (\geq 2hasGrape) \sqcap (\mathbf{prim}\ Wine)$ (i.e., why the set of good wines that are made from three or more grape types is subsumed by the set of wines that are made from two or more grape types). In this proof, the applications of rules **Eq** (i.e., equivalence) and **AndEq** (i.e., equivalence with a conjunction expression) are trivial, and should be removed.

Based on the resulting proofs, McGuinness's strategy to explain an entailment of the form $C \sqsubseteq D$, where C and D are class expressions, is as follows: first find two descriptions $C_1 \sqcap \ldots \sqcap C_m$ equivalent to C and $D_1 \sqcap \ldots \sqcap D_n$ equivalent to D, then find some C_j $(1 \leq j \leq m)$ such that C_j is subsumed by every D_i $(1 \leq i \leq n)$. From the proof in Table 3.2, for instance, the following explanation is generated:[1]

$$A \sqsubseteq (\mathbf{prim}Wine) \text{ because } \mathbf{Prim}(\tilde{\theta} = \{Wine, Good\}, \tilde{\alpha} = \{Wine\})$$

$$A \sqsubseteq (\geq 2hasGrape) \text{ because } \mathbf{AtLst}(n = 3, m = 2, \pi = hasGrape)$$

Although the above output provides only a logical foundation (but not the final presentation) of the explanation, it explains the subsumption in the example in a relatively natural way. In the generation of such an output, the decisions on which inference steps in the input proof should be removed are mostly based on the author's intuition. More importantly, the proposed approach is tightly coupled with the execution of the structural

[1] In the first statement, $(\mathbf{prim}\ Good\ Wine) \sqsubseteq (\mathbf{prim}\ Wine)$ because $\{Wine\} \subseteq \{Wine, Good\}$ (according to the rule **Prim** in Table 3.1).

Table 3.3: The tableau proof generated by Borgida et al.'s system (reproduced from [BFH00]) to explain the subsumption $\exists hasFriend.\top \sqcap \forall hasFriend.\neg(\exists hasChild.\neg Doctor \sqcup \exists hasChild.Lawyer)) \sqsubseteq \exists hasFriend.\forall hasChild.(Rich \sqcup Doctor)$; the modal logic-based inference rules associated with each step are shown on the left

$$\exists hasFriend.\top \sqcap \forall hasFriend.\neg((\exists hasChild.\neg Doctor) \sqcup (\exists hasChild.Lawyer)) \vdash \exists hasFriend.(\forall hasChild.(Rich \sqcup Doctor))$$

$l\wedge$ (1)

$$\exists hasFriend.\top, \forall hasFriend.\neg((\exists hasChild.\neg Doctor) \sqcup (\exists hasChild.Lawyer)) \vdash \exists hasFriend.(\forall hasChild.(Rich \sqcup Doctor))$$

$l\Diamond$ (2)

$$\top, \neg((\exists hasChild.\neg Doctor) \sqcup (\exists hasChild.Lawyer)) \vdash \forall hasChild.(Rich \sqcup Doctor)$$

$l\neg\vee$ (3)

$$\top, \neg(\exists hasChild.\neg Doctor), \neg(\exists hasChild.Lawyer) \vdash \forall hasChild.(Rich \sqcup Doctor)$$

$r\Box$ (4)

$$\neg\neg Doctor, \neg Lawyer, \vdash Rich \sqcup Doctor$$

$l\neg\neg$ (5)

$$Doctor, \neg Lawyer, \vdash Rich \sqcup Doctor$$

$r\vee$ (6)

$$Doctor, \neg Lawyer, \vdash Rich \sqcup Doctor$$

$=$ (7)

TRUE

subsumption algorithm, and thus is limited to only inferences that can be computed by this algorithm.

Borgida et al. [BFH00] follow the thought of coupling explanations with the execution of the reasoning algorithm, but rely on a tableau-based algorithm for $\mathcal{ALC}$ [SS91]—a much more expressive description logic than the one in CLASSIC. They produce *tableau proofs* that explain how the tableau-based algorithm proves subsumption between two class expressions. To generate such proofs, they first modify the implementation of the tableau-based algorithm to make the transformation of $C \sqsubseteq D$ to $(C \sqcap \neg D) \sqsubseteq \bot$ traceable, then construct a *sequent calculus*—i.e., a set of inference rules based on the multi-modal propositional logic $K_{(m)}$ in which each rule corresponds to one or more steps in the execution of the modified tableau-based algorithm—to construct tableau proofs. Resulting proofs thus consist of a number of inference steps (each of which associates with a modal logic-based inference rule) that are *parallel* to main steps in the execution of the tableau-based algorithm. A key advantage of this approach is that the transformation from subsumption to unsatisfiability is hidden, so the original subsumption problem is preserved throughout. Table 3.3 shows an example of a tableau proof generated by their system.

As pointed out by the authors [BFH00], tableau proofs, such as the one in Table 3.3,

Table 3.4: The explanation generated by Borgida et al.'s system (reproduced from [BFH00]) based on the tableau proof in Table 3.3

On the lhs, $\exists hasFriend.\top$ can be strengthened with $\forall$-restrictions on role $hasFriend$ to yield $\exists hasFriend.(\top \sqcap \neg((\exists hasChild.\neg Doctor) \sqcup (\exists hasChild.Lawyer)))$. The subsumption can now be proven by showing that $\exists$-restrictions for role $hasFriend$ subsume. To prove $\exists hasFriend.X \sqsubseteq \exists hasFriend.Y$, it is sufficient to prove $X \sqsubseteq Y$. So in this case we are reduced to showing $\top \sqcap \neg((\exists hasChild.\neg Doctor) \sqcup (\exists hasChild.Lawyer)) \sqsubseteq \forall hasChild.(Rich \sqcup Doctor)$.
On the lhs, apply de Morgan's laws to propagate in negation, to get $(\forall hasChild.\neg\neg Doctor)$. To prove $\forall hasChild.X \sqsubseteq \forall hasChild.Y$, it it sufficient to show $X \sqsubseteq Y$. So in this case we are reduced to showing $\neg\neg Doctor \sqsubseteq (Rich \sqcup Doctor)$.
Double negation elimination on the lhs leaves $Doctor$.
The subsumption now follows because the description $Doctor$ is subsumed by $Doctor$, and the rhs is an expansion of $Doctor$, since it is a disjunction.

still include *irrelevant* steps for explanation purposes—e.g., the steps associating with the rules $l\wedge$ (which replaces conjunctions on the antecedent side with commas), and $l\neg\neg$ (which eliminates double negation on the antecedent side) in Table 3.3. Additionally, the modal logic-based inference rules, unlike McGuinness's deduction rules, are hard to understand—e.g., the rule $r\Box$ in Table 3.3 which extracts class expressions from universal quantifications on both antecedent and succedent sides. Therefore, Borgida et al. simplify such proofs (by removing irrelevant proof fragments), and construct *explanation rules*—i.e., simpler variants of the proposed modal logic-based rules—to present in surface explanations. Finally, "English templates" are used to generate surface explanations. As an example, the surface explanation this system generates from the tableau proof in Table 3.3 is shown in Table 3.4. This explanation is nevertheless still long and very technical. Moreover, the correlation with the execution of the tableau-based algorithm makes it more suitable for "proof checking" rather than "proof understanding" for the readers [HPS09a].

Borgida et al.'s approach [BFH00] is applied and implemented in the ontology editor OntoTrack [LN04]. On the basis of Borgida et al.'s work [BFH00], Kwong [Kwo05] investigates generating accessible explanations for subsumption entailments in $\mathcal{ALC}$ [SS91]. Tableau proofs are taken as inputs to generate such explanations. The explanation Kwong's system generates for the tableau proof in Table 3.3 is shown in Figure 3.1. More recently, Borgida et al.'s work [BFH00] has been extended to explain subsumption entailments in DL-Lite [CDGL$^+$05] family of description logics [BCRM08].

—Explanation—

1 $\exists hasFriend.\top \sqcap \forall hasFriend.\neg(\exists hasChild.\neg Doctor \sqcup \exists hasChild.Lawyer) \sqsubseteq \exists hasFriend.\forall hasChild.(Rich \sqcup Doctor)$ is true,
because $\top \sqcap \neg(\exists hasChild.\neg Doctor \sqcup \exists hasChild.Lawyer) \sqsubseteq \forall hasChild.(Rich \sqcup Doctor)$ is true.
(LEFT-DIAMOND)

2 $\top \sqcap \neg(\exists hasChild.\neg Doctor \sqcup \exists hasChild.Lawyer) \sqsubseteq \forall hasChild.(Rich \sqcup Doctor)$ is true,
because $\top \sqcap \neg\exists hasChild.Lawyer \sqcap \neg\exists hasChild.\neg Doctor \sqsubseteq \forall hasChild.(Rich \sqcup Doctor)$ is true.
(LEFT-NOT-OR)

3 $\top \sqcap \neg\exists hasChild.Lawyer \sqcap \neg\exists hasChild.\neg Doctor \sqsubseteq \forall hasChild.(Rich \sqcup Doctor)$ is true,
because $\neg\exists hasChild.Lawyer \sqcap \neg\exists hasChild.\neg Doctor \sqsubseteq \forall hasChild.(Rich \sqcup Doctor)$ is true.
(SIMPLIFICATION)

4 $\neg\exists hasChild.Lawyer \sqcap \neg\exists hasChild.\neg Doctor \sqsubseteq \forall hasChild.(Rich \sqcup Doctor)$ is true,
because $\neg Lawyer \sqcap \neg\neg Doctor \sqsubseteq Rich \sqcup Doctor$ is true.
(RIGHT-BOX)

5 $\neg Lawyer \sqcap \neg\neg Doctor \sqsubseteq Rich \sqcup Doctor$ is true,
because $\neg Lawyer \sqcap Doctor \sqsubseteq Rich \sqcup Doctor$ is true.
(LEFT-NOT-NOT)

6 $\neg Lawyer \sqcap Doctor \sqsubseteq Rich \sqcup Doctor$ is true,
because a conjunction of ***Doctor*** on the left hand side is always more specific than ***Doctor*** whereas a disjunction of ***Doctor*** on the right hand side is always more general than ***Doctor***. Therefore this subsumption must hold.
(EQUIVALENCE)

Figure 3.1: The explanation generated by Kwong's system (taken from [Kwo05]) for the tableau proof in Table 3.3

3.1.2 Justification-Based Explanations

Justifications for entailments [Kal06] are the current dominant form of explanations for entailments of ontologies [Hor11]. They are conceptually simpler than reasoning-based proofs and explanations, and are directly linked to axioms that are asserted by developers. They are independent of both the implementation and execution of reasoning algorithms—i.e, follow the reconstructive paradigm—so no understanding of reasoning algorithms or proof calculi is required to understand them. With the proposal of effective algorithms for computing justifications, finding justifications for entailments has become a key explanation facility in many ontology development tools, including Protégé [KMR04, HPS08a], Swoop [KPS+06, Kal06], TopBraid Composer [Top], and the RaDON plug-in of the NeOn Toolkit [JHQ+09].

However, as discussed in Section 2.4, understanding how to get from a justification to an entailment is not always easy. Horridge et al. [HPS09b, Hor11] propose the computation of lemmas representing steps in inferring an entailment from a justification to guide users on understanding the inference. More precisely, they propose the construction of a proof,

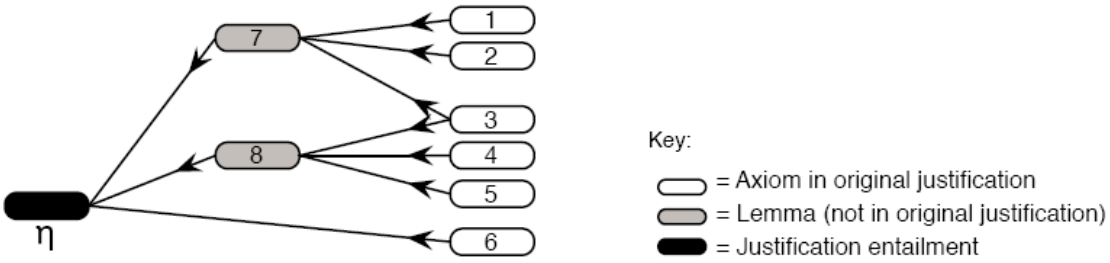


Figure 3.2: Schematic of a justification oriented proof (taken from [Hor11])

called a *justification oriented proof*, in which axioms in the justification are premises of the proof, lemmas are intermediate nodes, and the entailment is the root node. Unlike McGuinness's [McG96] and Borgida's [BFH00] proofs, justification oriented proofs are neither coupled with the execution of any reasoning algorithm, nor reliant on any proof calculus or deduction rules—i.e., follow the reconstructive paradigm. Every lemma in these proofs is an entailment drawn from a subset of premises and earlier lemmas—this subset is in fact a justification of the lemma. Figure 3.2 shows a schematic of a justification oriented proof. In this proof, the sets of axioms $\{1, 2, 3\}$, $\{3, 4, 5\}$, and $\{6, 7, 8\}$ are justifications for the lemmas 7 and 8, and the entailment η, respectively.

The understandability of such a proof depends upon the appropriateness of its lemmas; the more appropriate the lemmas, the more understandable the proof. This analysis leads to a research question of which lemmas most facilitate human understanding. Horridge et al. [HBPS11, Hor11] address this question by proposing a framework that can compute helpful lemmas to construct a justification oriented proof. This framework utilises the computation of the deductive closure of a set of axioms (i.e., the set of all entailments plus the original axioms) to compute candidate lemmas. Specifically, for a given entailment η and a justification J, this framework first computes the deductive closure of subsets of axioms in J. Since the deductive closure of a set of axioms is theoretically *infinite*, only entailments of the form $C \sqsubseteq D$ are computed, where C and D are class names or class expressions of fixed forms such as $\exists R_o.C$, $\forall R_o.C$, $\leq nR_o.C$, $\geq nR_o.C$, $\exists R_o.\{a\}$ etc.

From the computed deductive closure, all justifications for the entailment η are computed. The cognitive difficulty of these justifications is also assessed by means of a cognitive complexity model [HBPS11, Hor11] (this model will be described in detail in Section 7.1). Among the resulting justifications, only the easiest justification J' is considered. J' is returned if it is easier than J; otherwise, J is returned. This process is called *lemmatising*

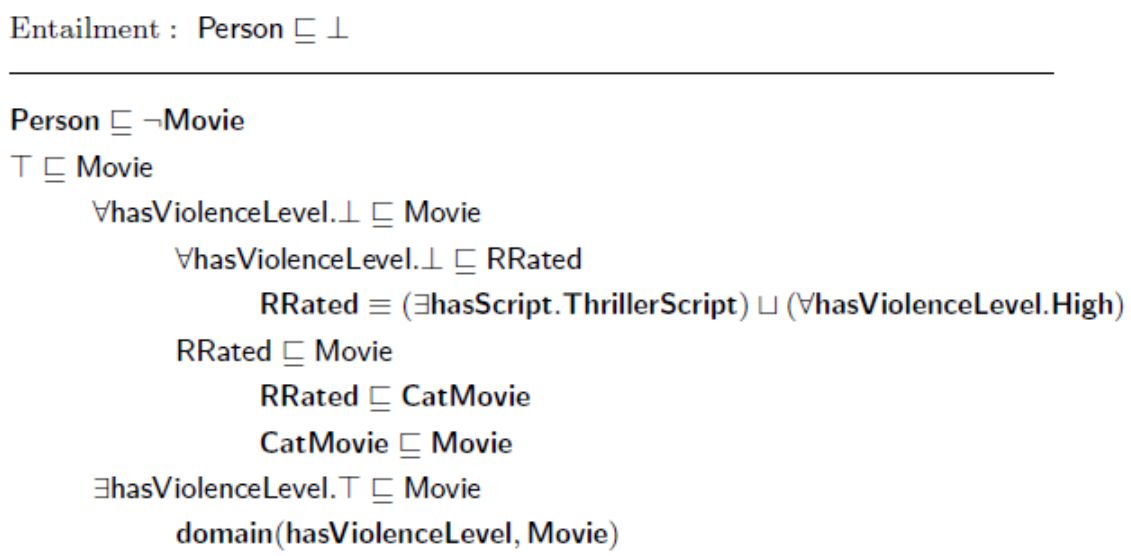


Figure 3.3: The justification oriented proof generated by Horridge's system (taken from [Hor11]) for the entailment $Person \sqsubseteq \bot$, with axioms at the terminal nodes (in bold) are axioms in the input justification

a justification. For the example proof in Figure 3.2, this process returns the justification $J' = \{6, 7, 8\}$.

The above-mentioned process is recursively repeated on axioms in J' until a complete proof is constructed. As an example, Figure 3.3 shows an justification oriented proof generated by Horridge at al.'s system to explain why the entailment $Person \sqsubseteq \bot$ follows from the justification $J = \{RRated \equiv (\exists hasScript.ThrillerScript) \sqcup (\forall hasViolenceLevel.High)$, $RRated \sqsubseteq CatMovie$, $CatMovie \sqsubseteq Movie$, and $\mathbf{Dom}(hasViolenceLevel, Movie)\}$.

As can be seen from Figure 3.3, justification oriented proofs appear closer to the way people do reasoning, more concise and readable than reasoning-based proofs. Moreover, the proof construction framework is generic as it relies on neither a proof calculus nor a set of deduction rules, so is capable of producing proofs for a number of justifications (even though the computational overhead may also be large due to the large number of calls to automated reasoners). However, the way this framework selects lemmas does not guarantee to produce the best proofs for human understanding. It selects only lemmas of some fixed class subsumption forms, regardless of the type of the original inference. Our corpus-based study (reported in Chapter 4) shows that many justifications include pairs of property-related axioms such as $\{R_p \sqsubseteq S_p, S_p \sqsubseteq T_p\}$, and $\{\mathbf{Tra}(R_p), \mathbf{Invs}(R_p, S_p)\}$. In such cases, none of the class subsumption lemmas would be helpful, but property-related lemmas such as $R_p \sqsubseteq T_p$ and $\mathbf{Tra}(S_p)$ would be.

The proposed cognitive complexity model [HBPS11, Hor11], as well as the way it is used in the proof construction framework, also has shortcomings. As will be explained in Section 7.1, this model [HBPS11] is biased towards measuring *structural complexity* of a justification rather than its cognitive difficulty. In the framework, this model is used locally to measure the difficulty of candidate justifications at each inference step, but neither the difficulty of the inference step itself, nor the difficulty of the overall proof, is measured. As a result, the output proof may not facilitate human understanding.

Finally, justification oriented proofs alone are not sufficient for users, especially non-expert users. As mentioned before, there exist naturally occurring inferences in OWL that are difficult to understand. An example of such inferences is the one to infer $\forall hasViolenceLevel.\bot \sqsubseteq RRated$ from $RRated \equiv (\exists hasScript.ThrillerScript) \sqcup (\forall hasViolenceLevel.High)$ in Figure 3.3. Our empirical study (reported in Section 7.2) shows that only 4% of participants could understand even a simplified version of this inference. A user study conducted by Horridge et al. [HPS09b, Hor11] also shows that many OWL experts have trouble in understanding justifications containing this inference. Such an inference should be identified, and further elucidation should be provided to help end-users, both experts and non-experts, understand it correctly.

3.2 Explaining Mathematical Theorems

The reconstructive paradigm [WT92] has also been applied in presenting and explaining mathematical proofs. Many researchers investigate transforming machined-generated proofs for mathematical theorems, in the form of resolution proofs [Pfe87, Lin90, And80, Mil84], into *natural deduction*[2] proofs [Gen35].

In Lingenfelder's work [Lin90], for instance, resolution proofs are first transformed into refutation graphs, then natural deduction proofs. Thereafter, natural deduction proofs are restructured to make them more readable for users. Specifically, given a natural deduction proof, trivial sub-proofs are replaced by trivial arguments, and non-trivial sub-proofs are replaced by lemmas. These lemmas are explained in separate proofs that are often presented before the main proof. Shared proofs are allowed to explain lemmas

[2]Natural deduction is a formalism in which logical reasoning is expressed by a set of inference rules that reflect the natural way of reasoning.

NNo	S;D		Formula	Reason
			Definitions	
1.	;1	$\vdash$	$\forall_u\ u * u^{-1} = e$	(L-Def-Inverse)
2.	;2	$\vdash$	$\forall_u\ e * u = u$	(L-Def-Unit)
			The Proof	
3.	;3	$\vdash$	$\forall_{x,y,z} x \in S \wedge y \in S \wedge x * y^{-1} = z \Rightarrow z \in S$	(Hyp)
4.	;4	$\vdash$	$a \in S$	(Hyp)
5.	;3	$\vdash$	$\forall_{y,z} a \in S \wedge y \in S \wedge a * y^{-1} = z \Rightarrow z \in S$	($\forall$E 3)
6.	;3	$\vdash$	$\forall_z a \in S \wedge a \in S \wedge a * a^{-1} = z \Rightarrow z \in S$	($\forall$E 5)
7.	;3	$\vdash$	$a \in S \wedge a \in S \wedge a * a^{-1} = e \Rightarrow e \in S$	($\forall$E 6)
8.	1;	$\vdash$	$a * a^{-1} = e$	($\forall$E 1)
9.	;4	$\vdash$	$a \in S \wedge a \in S$	($\wedge$I 4 4)
10.	1;4	$\vdash$	$a \in S \wedge a \in S \wedge a * a^{-1} = e$	($\wedge$I 9 8)
11.	1;3,4	$\vdash$	$e \in S$	($\Rightarrow$E 10 7)
12.	2;	$\vdash$	$e * a^{-1} = a^{-1}$	($\forall$E 2)
13.	;3	$\vdash$	$\forall_{y,z} e \in S \wedge y \in S \wedge e * y^{-1} = z \Rightarrow z \in S$	($\forall$E 3)
14.	;3	$\vdash$	$\forall_z e \in S \wedge a \in S \wedge e * a^{-1} = z \Rightarrow z \in S$	($\forall$E 13)
15.	;3	$\vdash$	$e \in S \wedge a \in S \wedge e * a^{-1} = a^{-1} \Rightarrow a^{-1} \in S$	($\forall$E 14)
16.	1,2;3,4	$\vdash$	$e \in S \wedge a \in S \wedge e * a^{-1} = a^{-1}$	($\wedge$I 11 4 12)
17.	1,2;3,4	$\vdash$	$a^{-1} \in S$	($\Rightarrow$E 16 15)
18.	1,2;3	$\vdash$	$a \in S \Rightarrow a^{-1} \in S$	($\Rightarrow$I 17)
19.	1,2;3	$\vdash$	$\forall\ x \in S \Rightarrow x^{-1} \in S$	($\forall$I 18)

Figure 3.4: An example of a natural deduction proof generated by Lingenfelder's system (taken from [Lin90])

that repeatedly occur in the original proof. Hence, the resulting proofs are more concise and easier to follow than the original proofs. An example of a natural deduction proof generated by Lingenfelder's system is shown in Figure 3.4.

Based on natural deduction proofs for mathematical theorems, several NLG systems have been built to translate them into natural language explanations; prominent examples being EXPOUND [Che76], MUMBLE [McD83], and THINKER [EP93]. The EXPOUND system [Che76] takes as inputs a natural deduction proof and lexical information of all predicates within the proof, and outputs an explanation in English. To generate such an explanation, the system first constructs a graph representing the inferential relationships between *lines* within the proof—each node in this graph consists of the identifier of a line, and the links show the inferential relationships between the lines. Based on the graph, it plans and generates an English explanation of how each line in the original proof is obtained from previous lines. When planning for the explanation, nodes in the graph are grouped to outline paragraphs, and trivial lines are ignored (but the decision of which lines should be ignored is based on the author's intuition). The generation of the English explanation is by means of "English templates". The formulation of an English template for a line depends entirely on the *local properties* of the line—i.e., its formula, the inference

rule used, and the premise lines in the discourse.

MUMBLE [McD81] is a general purpose system aiming to provide natural language interface for expert systems. It accepts as inputs various formats, including natural deduction proofs for mathematical theorems, and is capable of generating English explanations from these proofs [McD83]. As in the EXPOUND system [EP93], this system explains a natural deduction proof by showing how each line in the proof is related to earlier lines. However, the MUMBLE system enhances the output explanations by relying on the *contextual information* of each line—e.g., the choices of English text for earlier lines, and the difficulty of the associated inference step (again judged by the author's intuition)—to determine whether and how to translate it into English. Various NLG techniques, such as the pronominalisation of subsequent references, and the use of parentheses to cancel ambiguities, are also applied to make the output explanations more concise and coherent.

THINKER [EP93] is the first system that explores the generation of *multiple* English explanations from a natural deduction proof at different *levels of abstraction* and *styles* in order to serve for a wide range of users. It provides four distinct dimensions for users to decide what the output explanation will look like; they are: (1) the approach for the entire explanation (top-down, bottom-up, or mixture of the two), (2) the level of abstraction for the entire explanation (ranging from complete to a high-level overview), (3) the level of abstraction for explanations of individual inference steps (explaining only the inference rules, the inference rules plus the connectives, or complete explanations), and (4) when to produce the explanation (during or after the construction of the proof). The difference between the two options of the last dimension is in fact the style of the explanation—i.e., either it shows how the proof is constructed or it guides the readers to understand the proof.

One of the key drawbacks of the above-mentioned systems is that they rely on natural deduction proofs which in turn rely on inferences rules in the natural deduction calculus. Therefore, the resulting explanations often convey inference steps at too low a level of detail. Additionally, the decisions on which steps should be ignored are mostly based on the authors' intuition without having been empirically tested. Coscoy [Cos97] develops a similar system but relies on proofs of the *Calculus of Inductive Constructions*—a family of λ-calculi. However, this system also suffers similar problems.

NNo	S;D		Formula	Reason
			Definitions	
1.	;1	$\vdash$	$\forall_u\ u * u^{-1} = e$	(Def-Inverse)
2.	;2	$\vdash$	$\forall_u\ e * u = u$	(Def-Unit)
			The Proof	
3.	;3	$\vdash$	$\forall_{x,y,z} x \in S \wedge y \in S \wedge x * y^{-1} = z \Rightarrow z \in S$	(Hyp)
4.	;4	$\vdash$	$a \in S$	(Hyp)
5.	1;	$\vdash$	$a * a^{-1} = e$	(Def-Inverse)
6.	1;3,4	$\vdash$	$e \in S$	(3 4 4 5)
7.	2;	$\vdash$	$e * a^{-1} = a^{-1}$	(Def-Unit)
8.	1,2;3,4	$\vdash$	$a^{-1} \in S$	(3 6 4 7)
9.	1,2;3	$\vdash$	$a \in S \Rightarrow a^{-1} \in S$	($\Rightarrow$I 8)
10.	1,2;3	$\vdash$	$\forall\ x \in S \Rightarrow x^{-1} \in S$	($\forall$I 9)

Figure 3.5: The proof generated by Huang's system (taken from [Hua94]) from the natural deduction proof in Figure 3.4

Huang [Hua94] extends Lingenfelder's work by reconstructing natural deduction proofs to produce new proofs at a higher level of abstraction called *assertion level*. According to the author, at the assertion level, inference steps within a proof progress through the applications of either a given *premise*, or a predefined *theorem*. An example of such a theorem is "subset" [Hua94] in the set theory, which is encoded as: $U \subset F \leftrightarrow \forall x\ x \in U \rightarrow x \in F$ (U is a subset of F if and only if all instances of U are also instances of F).[3] In this way, resulting proofs are more concise, and resemble those found in mathematical textbooks. Figure 3.5 shows the natural deduction proof reconstructed by Huang's system from the proof in Figure 3.4 (generated by Lingenfelder's system).

To produce such a proof, Huang [Hua94] constructs a set of domain-specific compound inference rules called *assertion level* inference rules. The associated natural deduction proof of each assertion level rule—called the *natural expansion* of the rule—is also constructed. For a given input proof fed by Lingenfelder's system, all occurrences of natural expansions are searched and replaced by the associated assertion level rules, producing a new proof at the assertion level. This proof (in Figure 3.5) is shorter and more accessible than the input (in Figure 3.4).

Based on a assertion level proof, Huang [Hua94] develops a NLG system named PROVERB capable of generating an English explanation for the proof. This system employs a three-stage pipeline architecture [HF97], including *macro-planning*, *micro-planning*, and *realisation*. In these stages, sophisticated text planning techniques—such as *hierarchical planning* [Hov88, MP89, Dal92, HF97] (to split an explanation for a proof into a num-

[3]To prove the same inference requires a six-line natural deduction proof [Hua94].

Table 3.5: The explanation generated by Huang's system (reproduced from [HF97]) for the theorem: "Let $\mathcal{F}$ be a group, let $\mathcal{U}$ be a subgroup of $\mathcal{F}$, and let 1 and $1_{\mathcal{U}}$ be unit elements of $\mathcal{F}$ and $\mathcal{U}$. Then $1_{\mathcal{U}}$ equals 1."

Let $\mathcal{F}$ be a group, let $\mathcal{U}$ be a subgroup of F, and let 1 and $1_{\mathcal{U}}$ be unit elements of $\mathcal{F}$ and $\mathcal{U}$.
Because $1_{\mathcal{U}}$ is an unit element of $\mathcal{U}$, $1_{\mathcal{U}} \in \mathcal{U}$. Therefore, there is x such that $x \in \mathcal{U}$.
Let u_1 be such an x. Since $u_1 \in \mathcal{U}$ and $1_{\mathcal{U}}$ is a unit element of $\mathcal{U}$, $u_1 * 1_{\mathcal{U}} = u_1$. Since $\mathcal{F}$ is a group, $\mathcal{F}$ is a semigroup. Since $\mathcal{U}$ is a subgroup of $\mathcal{F}$, $\mathcal{U} \subset \mathcal{F}$. Because $\mathcal{U} \subset \mathcal{F}$ and $1_{\mathcal{U}} \in \mathcal{U}$, $1_{\mathcal{U}} \in \mathcal{F}$. Similarly, because $u_1 \in \mathcal{U}$ and $\mathcal{U} \subset \mathcal{F}$, $u_1 \in \mathcal{F}$. Then, $1_{\mathcal{U}}$ is a solution of $u_1 * x = u_1$.
Because $u_1 \in \mathcal{F}$ and 1 is an unit element of $\mathcal{F}$, $u_1 * 1 = u_1$. Since 1 is an unit element of $\mathcal{F}$, $1 \in \mathcal{F}$. Then, 1 is a solution of $u_1 * x = u_1$. Therefore, $1_{\mathcal{U}}$ equals 1. This conclusion is independent of the choice of u_1.

ber of explanations for sub-proofs within the original one), *local navigation* [HF97] (to group and present all lines about a particular object before turning to new objects), and text structure [Met92] (to bridge the gap between "what is to be said" and "how it is to be said")—and other NLG techniques—such as paraphrasing and aggregation—are applied to improve the fluency of explanations. As an example, Table 3.5 shows an English explanation generated by the PROVERB system.

Horacek [Hor99] argues that in many cases Huang's proofs still contain trivial inference steps while provide inadequate explanations for complex inference steps. Hence, he proposes various kinds of enhancements. In particular, each assertion level rule that includes implicit applications of modus tollens—a rule of inference in propositional logic which infers $\neg P$ from $P \rightarrow Q$ and $\neg Q$, and is found to be not easy for human understanding [JLB91]—is replaced by a new rule in which the last application of modus tollens is made explicit to the readers. This is done by first expanding the rule to a natural deduction proof, based on which the new rule is formulated.

Figure 3.6 shows an instance of such a formulation: the original complex rule (marked as (1) in the figure) is first expanded to the natural deduction proof (marked as (3)), from which the new rule (marked as (2)) is formulated. Consequently, the inference of $\neg(aRb)$ from (bRc) and $\neg(aRc)$ for a transitive relation R is explained more clearly as follows: "$\neg(aRc)$ implies $\neg(aRb)$ or $\neg(bRc)$. Thus, (bRc) yields $\neg(aRb)$" [Hor99]. However, the identification of both complex rules and their replacement rules are based on the author's intuition. Another enhancement is the proposal of *presentation rules* [Hor98] to omit trivial inference steps as well as group multiple related steps in the input proof. As a result, proofs and explanations generated by the PROVERB system are improved, but as

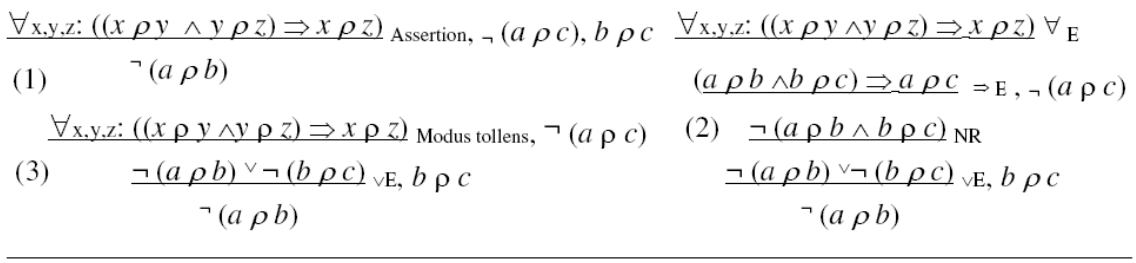


Figure 3.6: Reconstruction of a complex assertion level inference rule in Huang's work (taken from [Hor99])

pointed out by the author, are still far from satisfactory.

More recently, Fiedler [Fie05] develops a proof explanation system called **P.rex**, which operates in a "user-adaptive" paradigm. This system stores information about the mathematical knowledge and skills in *user models*, one for each user. In this way, it can retrieve the information of a particular user, and choose an appropriate level of abstraction for explanations presented to this user. Another distinctive feature of this system is that it allows users to interact with the system, and ask questions about the input proof in order to refine their understanding of the proof. Decisions on how to react to a user are also based on the knowledge and skills of the user. To model the mathematical knowledge and skills of a user, the **P.rex** system employs ACT-R (Adaptive Control of Thought-Rational) [AL98], a theory on cognitive architecture which divides knowledge in the world into two types: declarative knowledge (i.e., things or facts that we aware we know and can be memorised) and procedural knowledge (i.e., rules that we follow but we are not conscious of). However, the above-mentioned problems in presenting and explaining mathematical proofs have not been solved by this system.

3.3 Discussion and Conclusions

Explanations provided by McGuinness's [McG96] and Borgida et al.'s [BFH00] systems are reasoning-based, so are more suitable for 'checking' than 'understanding' the reasoning [Hor11]. In addition, understanding these explanations requires some knowledge of reasoning algorithms; therefore, they are more suitable for system developers rather than ordinary ontology developers.

Justification oriented proofs [Hor11] follow the reconstructive paradigm (discussed in Sec-

tion 2.4). They go beyond the advantages of justifications by introducing lemmas representing steps in inferring an entailment from a justification. They appear to be closer to the way people do reasoning, more concise and readable than reasoning-based explanations. However, further investigations are required to ensure the computation of good proofs and explanations that are helpful for non-expert users in understanding undesired entailments as well as debugging them. In particular, it is unknown which of various possible lemmas provides the best subdivision of the original inference into multiple steps, which inference steps are difficult to understand, and how to explain these inferences in a way that facilitates human understanding. Additionally, when explaining an undesired entailment, it is unknown how to make sources of errors best visible for end-users, as well as provide suggestions to repair them.

Research on presenting and explaining proofs for mathematical theorems suggests a number of ideas and techniques for planning and generating readable and user-oriented explanations. Specifically, proofs should be converted into a higher level of abstraction in which obvious inference steps are omitted, and the line of reasoning is made visible, before generating explanations. Moreover, different levels of abstraction and styles of explanation should be defined to serve a wide audience. Micro-planning techniques in NLG such as text structure [Met92, HF97] and aggregation rules [HF97] should be applied to improve the fluency of explanations. Investigations into whether (and how) these ideas and techniques can be applied to explaining entailments of OWL ontologies is, therefore, necessary.

Chapter 4

Construction of Deduction Rules

For our purposes, a *deduction rule* is a pattern comprising a set of premises and a conclusion (that can be drawn from the premises), abstracted so that they contain no entity names (except the two class names $\top$ and $\bot$), only variables. This chapter describes the construction of a set of deduction rules as generic representatives of basic inferences in OWL. These rules are based on *deduction patterns* computed from a large corpus of published real world OWL ontologies. Section 4.1 introduces the corpus, and explains why it is suitable for examining inferences in OWL. Sections 4.2 and 4.3 describe the computation of entailment-justification pairs and deduction patterns. Finally, the derivation of deduction rules from these patterns is reported in Section 4.4.

4.1 Ontology Corpus

417 published real world OWL ontologies were originally collected. They were *non-empty*, *distinct*, *consistent* (i.e., the assertions in their $ABox$ were consistent with respect to all the definitions in their $RBox$ and $TBox$), and *parsable* (i.e., containing no syntax errors) OWL ontologies collected from *three* different sources. Specifically, 190 ontologies were collected from the TONES repository [TON], 132 from the Ontology Design Patterns repository [Ontb], and remaining 95 from the Swoogle search engine [DFJ+04]. It should

be noted that these ontologies might nevertheless still contain other types of mistakes and semantic errors such as unsatisfiable classes (e.g., "Nothing is a movie").

The TONES repository [TON] is a database of OWL ontologies designed for use by system developers, especially in testing automated reasoners. Many ontologies in this database are biomedical ontologies collected from the NCBO BioPortal repository [RMKM08, NSW+09]—a repository of actively used ontologies in the biomedical community. Currently, the repository consists of 219 ontologies with a wide range of topics, expressiveness, and authoring styles. However, 29 of them are either empty, inconsistent, not parsable due to syntax errors, import ontologies that no longer exist, or duplicate other ontologies. None of these were included.

Ontology Design Patterns [Onta], a semantic web portal that consists of patterns often used in modelling ontologies, provides a database of real world OWL ontologies that are built by applying published ontology patterns [Ontb]. 204 ontologies were originally collected from this database; however, only 132 of them met our criteria, and so were selected.

Unlike TONES and Ontology Design Patterns, Swoogle [DFJ+04] is a search engine that allows people to search for RDF-based documents available on the Web. Using the tool, 150 published OWL ontologies were collected; but only 115 of them met our criteria, and so were selected.[1]

Since we were interested in subsumption inferences existing in OWL ontologies, only ontologies that had at least one non-trivial subsumption entailment were selected. The Pellet reasoner [SPG+07] was used to check whether an ontology had a non-trivial subsumption entailment. Details of the computation of subsumption entailments from each ontology will be described shortly in Section 4.2. As a result, 179 ontologies having at least one non-trivial subsumption entailment were selected as a corpus for our empirical study. Key features of this corpus that made it suitable for our purposes are as follows:

Large size It contains 179 ontologies, each of which has one or more non-trivial subsumption entailments. These ontologies contain over 200,000 axioms—large enough to assess commonly occurring inference patterns from OWL ontologies.

[1] The full list of ontologies in the corpus and their details can be found at `http://mcs.open.ac.uk/nlg/SWAT/index.html`.

Real world human-built It is composed mostly of real-world human-built ontologies, so reflecting the interests and habits of our target users.

Diverse in topic It covers a wide range of topics, ranging from biomedicine (such as the gene ontology), health care (such as the SNOMED ontology), to more familiar topics such as university, economy, country, etc.

Diverse in size and expressivity It contains ontologies of size 3 to 36,605 axioms, and the average size is 1,131. Their expressivity ranges from lightweight $\mathcal{EL}++$ to highly expressive $\mathcal{SHOIN}$, and even $\mathcal{SROIQ}$.

Diverse in authoring style It is authored by a wide range of human developers. Both named classes and deeply nested complex class expressions are used in most ontologies.

4.2 Collecting Justifications for Entailments

4.2.1 Method

The scope of this thesis is restricted to subsumption entailments belonging to the following three categories: (1) $\top \sqsubseteq A$, (2) $A \sqsubseteq \bot$, and (3) $A \sqsubseteq B$, where A and B are class names. This classification is useful for the examination of OWL inferences because it separates definitely undesirable entailments from uncertain ones. The first two categories indicate two types of problematic classes: (1) classes defined to be *semantically equivalent* to $\top$ (e.g., "Everything is a movie"), and (2) classes that are *unsatisfiable* (e.g., "Nothing is a person"). For the third category, the judgement of whether an entailment is undesirable must be based on domain knowledge. The entailment $Person \sqsubseteq Movie$ ("Every person is a movie"), for instance, is undesirable whereas the the entailment $Person \sqsubseteq Mammal$ ("Every person is a mammal") is not.

We developed a Java-based program to compute entailments and justifications from the corpus. In this program, the OWL-API package [HB09]—an open source application programming interface for OWL 2—was employed to parse the structure of OWL ontologies and axioms; the Pellet reasoner [SPG+07] was employed to compute all subsumption entailments from an ontology (including those that were already in the ontology); finally,

Table 4.1: Results of the computation of subsumption entailments and justifications with ten or fewer axioms, where 'A' and 'B' are class names

Entailment Type	**Entailment Number**	**Justification Frequency**	**Ontology Frequency**	**Pattern Number**
$\top \sqsubseteq A$	6 (0.0%)	28 (0.0%)	6 (3.5%)	9 (0.2%)
$A \sqsubseteq \bot$	572 (1.1%)	1,930 (1.3%)	32 (18.7%)	484 (8.2%)
$A \sqsubseteq B$	51,505 (98.9%)	151,850 (98.7%)	167 (97.7%)	5,415 (91.6%)
TOTAL	52,083 (100.0%)	153,808 (100.0%)	171 (100.0%)	5,908 (100.0%)

Horridge's program that implemented the algorithms described in [KPHS07] was employed to compute all justifications for an entailment. Our program sequentially computed all subsumption entailments and all their justifications from each ontology in the corpus.

Since both the computation of all subsumption entailments from an ontology and all justifications for an entailment were not guaranteed to be terminated within a reasonable time [KPHS07], especially for large ontologies, we set up a time out of *ten* minutes for each computation. Our program was deployed and run on a super-computer with a 64-bit Java virtual machine installed in order to maximise the number of entailments and justifications collected from the corpus.

4.2.2 Results

From the corpus, over 52,000 subsumption entailments of the three above-mentioned categories and approximately 222,000 non-empty and non-trivial justifications were collected. The size of the collected justifications varied from 1 up to 192 axioms, but 153,808 (or 69.4%) of them contained ten or fewer axioms. As a first attempt to examine inferences in OWL, this empirical study focused only on these justifications[2]. Details of these justifications are summarised in Table 4.1 ("Justification Frequency", "Ontology Frequency", and "Pattern Number" in this table will be explained in Section 4.3). This table shows that there was a very limited coverage of category 1 ($\top \sqsubseteq A$), a reasonable coverage of category 2 ($A \sqsubseteq \bot$), and an overwhelming preponderance of category 3 ($A \sqsubseteq B$).

[2]The analysis of larger justifications is a topic for future work.

Table 4.2: Example entailment-justification pairs and their deduction patterns

ID	Entailment and Justification	Pattern
1	Entailment: $WaterBody \sqsubseteq PlanetaryStructure$ Justification: 1.$PlanetaryStructure \equiv EarthRealm$ 2.$WaterBody \equiv BodyOfWater$ 3.$BodyOfWater \sqsubseteq EarthRealm$	Entailment: $C_0 \sqsubseteq C_1$ Justification: 1.$C_1 \equiv C_2$ 2.$C_0 \equiv C_3$ 3.$C_3 \sqsubseteq C_2$
2	Entailment: $DrySeasonDuration \sqsubseteq Occurrence$ Justification: 1.$Event \equiv Duration$ 2.$Event \equiv Occurrence$ 3.$DrySeasonDuration \sqsubseteq Duration$	Entailment: $C_0 \sqsubseteq C_1$ Justification: 1.$C_3 \equiv C_2$ 2.$C_3 \equiv C_1$ 3.$C_0 \sqsubseteq C_2$
3	Entailment: $LongWaveRadiation \sqsubseteq ElectromagWave$ Justification: 1.$ElectromagRadiation \equiv ElectromagWave$ 2.$InfraredRadiation \equiv LongWaveRadiation$ 3.$InfraredRadiation \sqsubseteq ElectromagRadiation$	Entailment: $C_0 \sqsubseteq C_1$ Justification: 1.$C_2 \equiv C_1$ 2.$C_3 \equiv C_0$ 3.$C_3 \sqsubseteq C_2$
4	Entailment: $PurkinjeFiber \sqsubseteq CardiacMuscle$ Justification: 1.$Myocardium \equiv CardiacMuscle$ 2.$ConductiveFibre \equiv PurkinjeFiber$ 3.$ConductiveFibre \equiv Myocardium \sqcap \exists function.Conduction$	Entailment: $C_0 \sqsubseteq C_1$ Justification: 1.$C_2 \equiv C_1$ 2.$C_3 \equiv C_0$ 3.$C_3 \equiv C_2 \sqcap \exists R_0.C_4$

4.3 Collecting Deduction Patterns

Justifications for OWL entailments can be very diverse, for four reasons. First, they differ in *material*—i.e., names of entities. Secondly, they can differ in *size*—i.e., the number of axioms. Although justification size is usually small compared with ontology size, a justification may contain dozens of axioms, or even more. Justifications computed from our corpus, for instance, have sizes varying from 1 up to 192 axioms. Thirdly, justifications can differ in the *logical structure of axioms*—i.e., containing different axioms. Finally, even when justifications have similar axioms, their *argument arrangements* in axioms may differ. The first three justifications in Table 4.2, for instance, have similar axioms but different argument arrangements ($C_1C_2 - C_0C_3 - C_3C_2$ in case 1, $C_3C_2 - C_3C_1 - C_0C_2$ in case 2, and $C_2C_1 - C_3C_0 - C_3C_2$ in case 3). Among these justifications, 1 and 2 are indeed two distinct justifications, but 1 and 3 are similar due to the commutative property of $\equiv$.

Recall that justifications provide a basis for generating explanations. However, given the diversity of justifications, there might be some doubt as to whether it is possible to find a generic set of rules to map them to rhetorical patterns in natural language.

On closer inspection, we nevertheless found that although many entailment-justification pairs looked different, they were actually *structurally equivalent*, and hence, should be mapped into a common *deduction pattern*—a pattern comprising a set of premises and a conclusion (that could be drawn from the premises), abstracted so that they contained no entity names (except the two class names $\top$ and $\bot$), only variables. In fact, our empirical study (reported here) showed that entailment-justification pairs computed from our corpus conformed to a smaller number of deduction patterns, and some deduction patterns were more frequent than others.

4.3.1 Structural Equivalence Relation

A justification can be considered as a minimal ontology. To the best of our knowledge, there has been no complete work on identifying the structural equivalence relation between two arbitrary justifications in OWL, and more generally, between two arbitrary OWL ontologies. The OWL 2 Structural Specification [OWL12c] defines a number of conditions to check whether any two objects in OWL (i.e., named entities, complex class and property expressions, and axioms) are structural equivalent. According to this specification, two class expressions, for instance, are structurally equivalent if they are composed of the same number and types of constructors, and the same named entities as arguments. This means that the difference in material is taken into account here. Hence, the class expression $(Person \sqcap \exists loves(Cat \sqcup Dog))$ is determined as structurally equivalent to $((\exists loves(Dog \sqcup Cat)) \sqcap Person)$, but not to $(Person \sqcap \exists loves(Cat \sqcup Rabbit))$.

Instead of the above-mentioned conditions, what are required here are looser conditions in which only the *logical structure* of expressions and axioms are taken into account, but not the named entities in their arguments, so that the three class expressions above can be determined as structurally equivalent. Horridge [Hor11] has recently provided a formal definition of *syntactic isomorphism* (i.e., structural equivalence) relation between any two δ-transformed $\mathcal{SHOIQ}$ axioms[3] based on a renaming method—that is, two axioms, α' and α'', are isomorphic if there is an injective renaming of each name in the signature of α' into a name in the signature of α''. The underlying idea of this method is to check

[3] δ-transformed $\mathcal{SHOIQ}$ axioms are those obtained after exhaustively applying the rewriting rules of the structural transformation δ [PG86, Hor11] on a set of $\mathcal{SHOIQ}$ axioms $\mathcal{S}$. The purpose of this transformation is to eliminate nested structures within axioms in $\mathcal{S}$, and convert $\mathcal{S}$ into an equi-consistent and equi-satisfiable set $\mathcal{S}'$ which consists of only "small and flat" axioms [Hor11].

whether the two axioms conform to a common logical structure. If there exists such a common structure, then the two axioms are structurally equivalent; otherwise, they are not.

The above-mentioned naming method alone, however, only works at the axiom level, and is not enough for identifying the structural equality between two entailment-justification pairs. Unlike an individual axiom, an entailment-justification pair as a whole forms a structure in which arguments (named entities) in both the entailment and the justification are interconnected. To check whether two entailment-justification pairs are structurally equivalent, not only the structural equivalence between individual axioms, but more importantly, the equivalence between arguments' internal interconnections need to be examined. The latter examination is in fact a generic and quite tricky problem. Building a graph representing both the logical structure and arguments' interconnections of each entailment-justification pair, then checking whether the two graphs are *isomorphic* is a possible solution, but its implementation is non-trivial, even for a much less expressive language than OWL [Sch88].

4.3.2 Method

Identifying the structural equivalence between any two entailment-justification pairs was not the main focus of our work. What we were interested here was a method for computing deduction patterns from entailment-justification pairs which was capable of mapping structural equivalent pairs into a common pattern. Our solution for this problem was to first compute a *canonical form* of each entailment-justification pair as its deduction pattern, then compare and collate the resulting deduction patterns to obtain a list of distinct deduction patterns.

The computation of canonical forms was done by substituting names of entities (i.e. classes, individuals, properties, datatypes, and literals) by alpha-numeric identifiers. Specifically, class names (except $\top$ and $\bot$) were replaced by C_0, C_1, ..., those of individuals by $i0$, $i1$, ..., those of object properties by R_{o0}, R_{o1}, ..., those of data properties by R_{d0}, R_{d1}, ..., those of datatypes by D_{t0}, D_{t1}, ..., and those of literal values by l_0, l_1, ... Table 4.2 shows some example deduction patterns for the associated entailment-justification pairs presented in the table.

The above substitution method had a limitation: it did not guarantee to always produce a single canonical form for structurally equivalent entailment-justification pairs, for two reasons. Firstly, OWL provides a number of *commutative* constructors not only for axioms (such as $\equiv$, **Dis**, **Invs**), but also for class expressions (such as $\sqcap$ and $\sqcup$). In such constructors, the arguments' order has no effect on the semantics, but its side effect is that it may result in a number of *variants* in logical structure of an axiom or expression, and so of a deduction pattern. For example, the deduction pattern 3 in Table 4.2 is a variant of pattern 1 caused by the occurrence of $\equiv$. Secondly, different orders in which entity names are substituted often result in different deduction patterns. For example, if the substitution of entity names in case 1 in Table 4.2 follows the order of "entailment-2-3-1" (instead of "entailment-1-2-3" as in the table) then the following deduction pattern—which looks different but is structurally equivalent to the one in the table—will be returned:

Entailment:
$C_0 \sqsubseteq C_1$
Justification:
1. $C_1 \equiv C_3$
2. $C_0 \equiv C_2$
3. $C_2 \sqsubseteq C_3$

To avoid duplications when computing deduction patterns, top-down alphabetical order was used every time there was an ordering option. In particular, two *sorting heuristics*—one for ordering axioms within a justification, and one for ordering arguments in a commutative constructor—were applied before and while performing the substitution of entity names. Given an entailment-justification pair, axioms in the justification were first sorted by their logical structure. This was done by converting the axioms into functional style strings [OWL12c] in which only constructors of all types were retained, then sorting the axioms based on the alphabetic order of these strings. Based on the entailment and the sorted justification, the deduction pattern was constructed, starting from the entailment.

The substitution of entity names was done by recursively parsing OWL constructors nested in each axiom. For each constructor, all of its arguments were analysed. If an argument was atomic, then it was replaced by an appropriate alpha-numeric identifier. Otherwise, it was recursively parsed. In this process, whenever a commutative constructor was encountered, its arguments were sorted by their logical structure before analysed. The result of this

process was a deduction pattern stored in a hash table in the form of a string. The pattern would be collated if a similar deduction pattern was found in the table. This collation was based on *string matching*.

The proposed method was nevertheless incomplete due to a limitation of the sorting heuristics: it would not provide a unique sorting if two or more axioms in the justification had the same logical structure. Consequently, the program might fail to produce a single pattern for two or more structural equivalent entailment-justification pairs. Examples of cases that might fail are the four cases in Table 4.2 because their justifications contain two $\equiv$ axioms. The deduction patterns of cases 1 and 3 are indeed structurally equivalent, and should be amalgamated. Therefore, after the computation of deduction patterns completed, we manually checked the resulting patterns in order to handcraft the failed cases—i.e., manually grouping structurally equivalent patterns that had *not* been collated yet. We started the manual handcraft from the most frequent patterns to the least frequent ones (i.e., having frequency of 1) in order to minimise the number of failed cases.

The result of the above computation was a list of distinct deduction patterns ranked by frequency. For each pattern, two types of frequencies were measured: (a) occurrences of the pattern across all ontologies (called *justification frequency*), and (b) the number of ontologies in which the pattern occurred at least once (called *ontology frequency*) (as in Table 4.1). Among the two measurements, ontology frequency was a more stable measure since it was relatively unaffected by the ontology size—i.e., a deduction pattern might occur in very few ontologies, but had a high justification frequency because these ontologies were very large. Therefore, it was used as the main frequency measurement of a deduction pattern.

4.3.3 Results

5,908 distinct deduction patterns were obtained in this way. Table A.1 in the Appendix A shows frequently occurring patterns of each entailment category computed from our corpus, sorted by ontology frequency then justification frequency. All nine patterns for $\top \sqsubseteq A$ entailments (where A is a class name) are shown in this table. For other entailment categories, ten patterns are shown, with the first three patterns being the most frequent ones. These patterns are relatively simple and quite similar to each other. Therefore, the

subsequent patterns were randomly selected in order to show the structural diversity of deduction patterns.

4.4 Collecting Deduction Rules

4.4.1 Method

From the computed deduction patterns, we manually formulated deduction rules that could serve as generic representatives of inference steps in an explanation. This required that these rules were *simple* and *frequent*. These requirements were used as our criteria in formulating the rules.

For *simplicity*, we required that deduction rules (a) contained no parts within their premises that were redundant with the conclusion, and (b) could not be divided into simpler deduction rules that were easier to understand. To achieve (a), the *laconic property* for deduction rules—that is, any premise within a rule should not contain information that is unnecessary for the entailment to hold—was considered. This property was based on Horridge's claim that superfluous parts in a justification might distract human understanding, and should be eliminated [HPS10, Hor11]. Therefore, we formulated two types of deduction rules, namely *single-premise* and *multiple-premise*. Single-premise rules were those that consisted of exactly one premise, and merely unpacked part of the meaning of the premise—the part that actually contributed to the entailment. On the other hand, multiple-premise rules were those that contained multiple premises, and none of the premises consisted of unnecessary information[4].

In assessing (b), we relied partly on intuition. As a rough guide, we assumed that a deduction rule ought to convey a reasonable and understandable amount of information so that the resulting explanation would not be too detailed or too brief. We assumed that any rule with over four premises would be too long to be understandable for most people, thus ought to be subdivided into simpler rules[5].

For *frequency*, we considered (a) the frequency of equivalent entailment-justification pairs

[4] An alternative approach was to formulate only multiple-premise deduction rules in which either *some* or *all* information from each premise was used to arrive at the conclusion. This approach would, however, result in a large number of structural variants of a rule due to the structural variety of the premises.

[5] The issue of understandability of deduction rules is re-examined in Chapter 7, which describes an empirical test of understandability of our deduction rules.

in the corpus, and (b) the frequency of more complex entailment-justification pairs that contained a deduction rule as a part. We manually examined the frequent deduction patterns computed from the corpus to find those that satisfied these criteria. Single-premise rules were formulated every time a premise in a deduction pattern contained a redundant part. The single-rule $\{X \equiv Y \sqcap Z\} \rightarrow X \sqsubseteq Y$ ('ObjInt-1'), for example, was formulated while examining the deduction pattern $\{C_0 \sqsubseteq C_2 \wedge C_2 \equiv C_1\} \rightarrow C_0 \sqsubseteq C_1$.

For multiple-premise rules, simple and frequent deduction patterns were first selected as deduction rules. An example of rules of this type is $\{X \sqsubseteq Y \wedge Y \sqsubseteq Z\} \rightarrow X \sqsubseteq Z$ ('SubCls-SubCls-1'), derived directly from the deduction pattern $\{C_0 \sqsubseteq C_2 \wedge C_2 \sqsubseteq C_1\} \rightarrow C_0 \sqsubseteq C_1$. This rule corresponded to the well-known syllogism that from "Every X is a Y" and "Every Y is a Z", we could infer "Every X is a Z", where X, Y, and Z were either arbitrary class names or complex class expressions.

Subsequently, further multiple-premise rules that occurred often as parts of more complex deduction patterns, but were not computed as separate patterns because of certain limitations of the reasoning algorithm[6], were added. For example, rule $\{X \sqsubseteq \forall R_0.Y \wedge \mathbf{Invs}(R_o, S_o)\} \rightarrow \exists S_o.X \sqsubseteq Y$ ('ObjAll-ObjInv') was derived while examining the following deduction pattern:

Entailment:
$C_0 \sqsubseteq \bot$
Justification:
1. $C_0 \sqsubseteq \exists r_0.C_1$
2. $\mathbf{Invs}(r_0, r_1)$
3. $\mathbf{Dis}(C_0, C_2)$
4. $C_1 \sqsubseteq \forall r_1.C_2$

The formulation of deduction rules was stopped when we collected enough rules to obtain an interesting range of explanations—in other words, covering most of the entailment-justification pairs studied from the corpus (i.e., at least 50%, preferably more).

[6]Reasoning services for OWL typically compute only some kinds of entailment, such as $A \sqsubseteq B$, $A \equiv B$, and $i \in A$ statements where A and B are class names and i is an individual name, and ignore others.

4.4.2 Results

So far, 57 deduction rules were obtained in this way, as listed in Table 4.3, each with a unique name. Among these rules, 11 of them (rules 1 to 11 in the table) were single-premise, and the remaining 46 were multiple-premise. For each rule, its basic form was first derived; thereafter, its *extended forms* were derived where possible, as shown in Table 4.3. These forms were in fact *structural variants* of the original rule—i.e., they had slightly different structures, but retained the key inference in the original rule[7]. For example, the simplest form of rule 'ObjInt-1' (rule 2 in Table 4.3) was $\{X \equiv Y \sqcap Z\} \rightarrow X \sqsubseteq Y$—e.g., from $Woman \sqsubseteq Person \sqcap Female$, it would follow that $Woman \sqsubseteq Person$. Based on this inference, the following extended form could be derived: $X \equiv \exists R_o.(Y \sqcap Z) \rightarrow X \sqsubseteq \exists R_o.Y$—e.g., from $DolphinLover \equiv \exists loves.(Mammal \sqcap Fish)$, it follows that $DolphinLover \sqsubseteq \exists loves.Mammal$.

Two-premise rules were the main kind of our multiple-premise rules (40 out of 46, or 87.0%) as they were minimum rules in which a new piece of information (presented in the entailment) was unpacked from the combination of multiple premises. In rule 'SubCls-DisCls' (rule 15 in Table 4.3), for instance, the conclusion $X \sqsubseteq \bot$ was inferred from the combination of two premises: $X \sqsubseteq Y$ and $\mathbf{Dis}(X, Y)$. If any of these premises was omitted, the conclusion would no longer be inferable. Some of our two-premise rules were directly derived from the two-premise deduction patterns computed from the corpus, such as rules 'DatMin-DatRng', 'DatVal-DatRng', and 'SubCls-SubCls-1' (rules 13, 14, and 39 in Table 4.3). Other two-premise rules were formulated from the analysis of the composition of larger deduction patterns, as described in the previous sub-section.

There were similarities between some of our rules and those defined in OWL 2 RL [OWL12b], a sub-language of OWL designed to support rule-based reasoning technologies. For example, rule 'ObjDom-SubCls' ($\{\mathbf{Dom}(R_o, X) \wedge X \sqsubseteq Y\} \rightarrow \mathbf{Dom}(R_o, Y)$) was exactly the same as rule 'scm-dom1' in the OWL 2 RL ($\{T(?P, rdfs:domain, ?C_1) \wedge T(?C_1, rdfs:subClassOf, ?C_2)\} \rightarrow T(?P, rdfs:domain, ?C_2)$ where P is an object property name). However, OWL 2 RL rules were designed to provide a useful basis for practical implementation of rule-based reasoning in OWL 2 RL, so many them were not suitable for

[7] An alternative approach was to consider the extended forms of a deduction rule as separate rules in their own right. However, we believed that the understandability of the extended forms would be similar to that of the original rule, so measuring their difficult level would be unnecessary. Therefore, considering them as the extensions of the original rule would be a better way.

explanation purposes. For example, rule 'prp-trp' in the OWL 2 RL—which says that from $T(?P, rdf:type, owl:TransitiveProperty)$ (P is transitive), $T(?x, ?P, ?y)$ (x is connected by P to y), and $T(?y, ?P, ?z)$ (y is connected by P to z), it follows that $T(?x, ?P, ?z)$ (x is connected by P to z) where P is an object property name, x, y, and z are individual names—was designed at the individual level, but what we needed here was a more general rule that worked at the class level such as rule 'ObjSom-ObjSom-ObjTra' in Table 4.3 ($\{X \sqsubseteq \exists R_o.Y \wedge Y \sqsubseteq \exists R_o.Z \wedge \mathbf{Tra}(R_o)\} \rightarrow X \sqsubseteq \exists R_o.Z$), or even a more compact rule such as 'ObjSom-ObjTra' in Table 4.3 ($\{X \sqsubseteq \exists R_o.(\exists R_o Y) \wedge \mathbf{Tra}(R_o)\} \rightarrow X \sqsubseteq \exists R_o.Y$). Therefore, we decided not to use OWL 2 RL rules, and formulated our own rules that were more suitable for explanation purposes.

In order to produce concise (i.e., having a small number of inference steps) explanations for entailments, deduction rules with three premises were also derived. It was true that three-premise rules were often harder to understand than two-premise rules, and that there might exist ways to subdivide them into multiple steps. For instance, instead of formulating rule 'DisCls-SubCls-SubCls' (rule 52 in Table 4.3) which yielded $\mathbf{Dis}(U, V)$ from three premises including $\mathbf{Dis}(X, Y)$, $U \sqsubseteq X$, and $V \sqsubseteq Y$, one could first group the first two premises to yield $\mathbf{Dis}(U, Y)$, then combined this statement with the third premise to yield the entailment. Similarly, instead of formulating rule 'SubCls-SubCls-DisCls' (rule 53 in Table 4.3) which derived $X \sqsubseteq \bot$ from $X \sqsubseteq Y$, $X \sqsubseteq Z$, and $\mathbf{Dis}(Y, Z)$, one could group the first and last premises to first yield $\mathbf{Dis}(X, Z)$, then combined this statement with the second premise to yield the entailment. However, we believed that such subdivisions were unnecessary as they added little or no help for human understanding, but lengthened the resulting proofs. Of course, this initial belief was based on our intuition, and an empirical test of understandability of the deduction rules would be necessary—if it turned out that a rule was too difficult for people to follow, then it would be necessary to subdivide it into simpler rules[8]. However, at this step, our intuition-based judgement of the added value of lemmas was used as a criterion to decide whether to formulate a three-premise deduction rule.

A total of five three-premise deduction rules were derived in this way (rules 52 to 56 in Table 4.3). These rules were either collected from the three-premise deduction patterns computed from the corpus (such as rule 'SubCls-SubCls-DisCls'), or formulated from the

[8]In fact, a study was later conducted to empirically measure the understandability of our deduction rules. It will be reported in detail in Chapter 7.

analysis of more complex deduction patterns (such as rule 'SubObj-ObjInv-ObjInv'). The only four-premise rule 'ObjVal-ObjVal-DifInd-ObjFun' (rule 57 in Table 4.3) was a special case. It was similar to rule 'DatVal-DatVal-DatFun', but was applied on an object property (instead of a data property as in rule 'DatVal-DatVal-DatFun'), so required four premises.

4.4.3 Coverage of the Rule Set

To assess the efficiency of the collected rule set, it is necessary to measure its *coverage* in the construction of proof trees from an entailment-justification pair—that is, from how many justifications for entailments from real world OWL ontologies it can construct at least a proof tree. We tested the rule set through the ontology corpus described in Section 4.1, but restricted to only justifications of size ten or less. The algorithm described in Chapter 5 was employed to construct proof trees. Details of the set-up of this test are discussed in Section 5.3.

The result of this test is summarised in Table 5.1. The test showed that our rule set was generic enough to generate proof trees, and so explanations, for 75.6% (or 116,318) of the input justification-entailment pairs. Among over 37,000 cases failed, over 97 percent of them were caused by the limited coverage of the rule set. Even so, these results have confirmed that our rule set covers most of the basic inferences that frequently occur in published real world OWL ontologies, so is efficient for the generation of explanations for subsumption entailments of OWL ontologies.

4.5 Conclusions and Future Work

We have performed an extensive corpus study (over 150,000 justifications from 171 ontologies) for a well-defined set of entailment types (three categories), and shown that we can find a surprisingly small set of simple deduction rules that covers most of them. Because they are *simple*, these rules are good candidates for understandable inference steps; because there are not many of them, we can investigate them within the practical limitations of a PhD project while still obtaining quite good coverage.

Part of the future work is to extend the current rule set to cover more inferences in OWL. Additionally, deduction rules at a higher level of abstraction such as those that group a

Table 4.3: Deduction rules collected through our empirical study

Size	ID	Name	Deduction Rule	Extended Forms
1	1	EquCls	$X \equiv Y[\equiv \ldots]$ $\rightarrow X \sqsubseteq Y$	
	2	ObjInt-1	$X \equiv Y_1 \sqcap \ldots \sqcap Y_n \sqcap Z_1 \sqcap \ldots \sqcap Z_m,\ n \geq 1, m \geq 1$ $\rightarrow X \sqsubseteq Y_1 \sqcap \ldots \sqcap Y_n$	$X \equiv \exists R_o.(Y_1 \sqcap \ldots \sqcap Y_n \sqcap Z_1 \sqcap \ldots \sqcap Z_m),\ n \geq 1, m \geq 1$ $\rightarrow X \sqsubseteq \exists R_o.(Y_1 \sqcap \ldots \sqcap Y_n)$
	3	ObjInt-2	$X \sqsubseteq Y_1 \sqcap \ldots \sqcap Y_n \sqcap Z_1 \sqcap \ldots \sqcap Z_m,\ n \geq 1, m \geq 1$ $\rightarrow X \sqsubseteq Y_1 \sqcap \ldots \sqcap Y_n$	$X \sqsubseteq \exists R_o.(Y_1 \sqcap \ldots \sqcap Y_n \sqcap Z_1 \sqcap \ldots \sqcap Z_m),\ n \geq 1, m \geq 1$ $\rightarrow X \sqsubseteq \exists R_o.(Y_1 \sqcap \ldots \sqcap Y_n)$
	4	ObjUni-1	$X \equiv Y_1 \sqcup \ldots \sqcup Y_n \sqcup Z_1 \sqcup \ldots \sqcup Z_m,\ n \geq 1, m \geq 1$ $\rightarrow Y_1 \sqcup \ldots \sqcup Y_n \sqsubseteq X$	$X \equiv \exists R_o.(Y_1 \sqcup \ldots \sqcup Y_n \sqcup Z_1 \sqcup \ldots \sqcup Z_m),\ n \geq 1, m \geq 1$ $\rightarrow \exists R_o.(Y_1 \sqcup \ldots \sqcup Y_n) \sqsubseteq X$
	5	ObjUni-2	$Y_1 \sqcup \ldots \sqcup Y_n \sqcup Z_1 \sqcup \ldots \sqcup Z_m \sqsubseteq X,\ n \geq 1, m \geq 1$ $\rightarrow Y_1 \sqcup \ldots \sqcup Y_n \sqsubseteq X$	$\exists R_o.(Y_1 \sqcup \ldots \sqcup Y_n \sqcup Z_1 \sqcup \ldots \sqcup Z_m) \sqsubseteq X,\ n \geq 1, m \geq 1$ $\rightarrow \exists R_o.(Y_1 \sqcup \ldots \sqcup Y_n) \sqsubseteq X$
	6	ObjExt	$X \sqsubseteq\!= n_1 R_o.Y,\ n_1 \geq n_2 \geq 0$ $\rightarrow X \sqsubseteq\!\geq n_2 R_o.Y$	$X \sqsubseteq\!= nR_o.Y,\ n \geq 0$ $\rightarrow X \sqsubseteq\!\leq nR_o.Y$ $X \sqsubseteq\!\geq n_1 R_o.Y,\ n_1 \geq n_2 \geq 0$ $\rightarrow X \sqsubseteq\!\geq n_2 R_o.Y$
	7	ObjAll	$X \equiv \forall R_o.Y$ $\rightarrow \forall R_o.\bot \sqsubseteq X$	
	8	Top	$\top \sqsubseteq X$ $\rightarrow Y \sqsubseteq X$	
	9	Bot	$X \sqsubseteq \bot$ $\rightarrow X \sqsubseteq Y$	
	10	ObjCom-1	$X \sqsubseteq \neg X$ $\rightarrow X \sqsubseteq \bot$	
	11	ObjCom-2	$\neg X \sqsubseteq Y$ $\rightarrow \top \sqsubseteq X \sqcup Y$	
2	12	DatSom-DatRng	$X \sqsubseteq \exists R_d.D_{r0}$ $\wedge\ \mathbf{Rng}(R_d, D_{r1})$, D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq \exists R_o.(\exists R_d.D_{r0})$ $\wedge\ \mathbf{Rng}(R_d, D_{r1})$, D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$
	13	DatMin-DatRng	$X \sqsubseteq\!\geq nR_d.D_{r0},\ n > 0$ $\wedge\ \mathbf{Rng}(R_d, D_{r1})$, D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq \exists R_o.(\exists R_d.D_r),\ n > 0$ $\wedge\ \mathbf{Rng}(R_d, D_{r1})$, D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$
	14	DatVal-DatRng	$X \sqsubseteq \exists R_d.\{l0 \star D_t\})$ $\wedge\ \mathbf{Rng}(R_d, D_r)$, D_t & D_r are disjoint $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq \exists R_o.(\exists R_d.\{l0 \star D_t\})$ $\wedge\ \mathbf{Rng}(R_d, D_r)$, D_t & D_r are disjoint $\rightarrow X \sqsubseteq \bot$
	15	SubCls-DisCls	$X \sqsubseteq Y$ $\wedge\ \mathbf{Dis}(X, Y[, \ldots])$ $\rightarrow X \sqsubseteq \bot$	
	16	Top-DisCls	$\top \sqsubseteq Y$ $\wedge\ \mathbf{Dis}(X, Y[, \ldots])$ $\rightarrow X \sqsubseteq \bot$	
	17	ObjMin-ObjMax	$X \sqsubseteq\!\geq n_1 R_o.Y$ $\wedge\ X \sqsubseteq\!\leq n_2 R_o.Y,\ 0 \leq n_2 < n_1$ $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq\!= n_1 R_o.Y$ $\wedge\ X \sqsubseteq\!\leq n_2 R_o.Y,\ 0 \leq n_2 < n_1$ $\rightarrow X \sqsubseteq \bot$
	18	ObjMin-ObjFun	$X \sqsubseteq\!\geq nR_o.Y,\ n > 1$ $\wedge\ \mathbf{Fun(R_o)}$ $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq\!= nR_o.Y,\ n > 1$ $\wedge\ \mathbf{Fun(R_o)}$ $\rightarrow X \sqsubseteq \bot$
	19	DatMin-DatFun	$X \sqsubseteq\!\geq nR_d.D_r,\ n > 1$ $\wedge\ \mathbf{Fun(R_d)}$ $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq\!= nR_d.Y,\ n > 1$ $\wedge\ \mathbf{Fun(R_d)}$ $\rightarrow X \sqsubseteq \bot$
	20	ObjSom-Bot-1	$X \sqsubseteq \exists R_o.Y$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq\!\geq nR_o.Y,\ n > 0$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$ $X \sqsubseteq\!= nR_o.Y,\ n > 0$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$

Continued on Next Page...

	21	ObjSom-Bot-2	$X \sqsubseteq \exists R_o.(Y \sqcap Z[\sqcap \ldots])$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq\geq nR_o.(Y \sqcap Z[\sqcap \ldots]),\ n > 0$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$ $X \sqsubseteq= nR_o.(Y \sqcap Z[\sqcap \ldots]),\ n > 0$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$
	22	ObjInt-DisCls	$X \sqsubseteq \exists R_o.(Y_1 \sqcap \ldots \sqcap Y_m),\ m \geq 2$ $\wedge\ \mathbf{Dis}(Y_1, \ldots, Y_m[, \ldots])$ $\rightarrow X \sqsubseteq \bot$	$X \sqsubseteq\geq nR_o.(Y_1 \sqcap \ldots \sqcap Y_m),\ n > 0, m \geq 2$ $\wedge\ \mathbf{Dis}(Y_1, \ldots, Y_m[, \ldots])$ $\rightarrow X \sqsubseteq \bot$ $X \sqsubseteq= nR_o.(Y_1 \sqcap \ldots \sqcap Y_m),\ n > 0, m \geq 2$ $\wedge\ \mathbf{Dis}(Y_1, \ldots, Y_m[, \ldots])$ $\rightarrow X \sqsubseteq \bot$
	23	SubCls-ObjCom-1	$X \sqsubseteq Y$ $\wedge\ X \sqsubseteq \neg Y$ $\rightarrow X \sqsubseteq \bot$	
	24	SubCls-ObjCom-2	$X \sqsubseteq Y$ $\wedge\ \neg X \sqsubseteq Y$ $\rightarrow \top \sqsubseteq Y$	
	25	ObjDom-ObjAll	$\mathbf{Dom}(R_o, X)$ $\wedge\ \forall R_o.\bot \sqsubseteq X$ $\rightarrow \top \sqsubseteq X$	$\exists R_o.\top \sqsubseteq X$ $\wedge\ \forall R_o.\bot \sqsubseteq X$ $\rightarrow \top \sqsubseteq X$
	26	SubObj-SubObj	$R_o \sqsubseteq S_o$ $\wedge\ S_o \sqsubseteq T_o$ $\rightarrow R_o \sqsubseteq T_o$	
	27	ObjTra-ObjInv	$\mathbf{Tra}(R_o)$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Tra}(S_o)$	
	28	ObjDom-SubCls	$\mathbf{Dom}(R_o, X)$ $\wedge\ X \sqsubseteq Y$ $\rightarrow \mathbf{Dom}(R_o, Y)$	
	29	ObjDom-SubObj	$\mathbf{Dom}(R_o, X)$ $\wedge\ S_o \sqsubseteq R_o$ $\rightarrow \mathbf{Dom}(S_o, X)$	
	30	ObjRng-ObjInv	$\mathbf{Rng}(R_o, X)$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Dom}(S_o, X)$	
	31	ObjRng-ObjSym	$\mathbf{Rng}(R_o, X)$ $\wedge\ \mathbf{Sym}(R_o)$ $\rightarrow \mathbf{Dom}(R_o, X)$	
	32	ObjRng-SubCls	$\mathbf{Rng}(R_o, X)$ $\wedge\ X \sqsubseteq Y$ $\rightarrow \mathbf{Rng}(R_o, Y)$	
	33	ObjRng-SubObj	$\mathbf{Rng}(R_o, X)$ $\wedge\ S_o \sqsubseteq R_o$ $\rightarrow \mathbf{Rng}(S_o, X)$	
	34	ObjDom-ObjInv	$\mathbf{Dom}(R_o, X)$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Rng}(S_o, X)$	
	35	ObjDom-ObjSym	$\mathbf{Dom}(R_o, X)$ $\wedge\ \mathbf{Sym}(R_o)$ $\rightarrow \mathbf{Rng}(R_o, X)$	
	36	ObjSom-ObjDom	$X \sqsubseteq \exists R_o.Z$ $\wedge\ \mathbf{Dom}(R_o, Y)$ $\rightarrow X \sqsubseteq Y$	$X \sqsubseteq\geq R_o.Z,\ n > 0$ $\wedge\ \mathbf{Dom}(R_o, Y)$ $\rightarrow X \sqsubseteq Y$ $X \sqsubseteq= nR_o.Z,\ n > 0$

Continued on Next Page...

			$\wedge\ \mathbf{Dom}(R_o, Y)$ $\rightarrow X \sqsubseteq Y$
37	DatSom-DatDom	$X \sqsubseteq \exists R_d.D_r$ $\wedge\ \mathbf{Dom}(R_d, Y)$ $\rightarrow X \sqsubseteq Y$	$X \sqsubseteq\!\geq R_d.D_r,\ n > 0$ $\wedge\ \mathbf{Dom}(R_d, Y)$ $\rightarrow X \sqsubseteq Y$ $X \sqsubseteq\!= nR_d.D_r,\ n > 0$ $\wedge\ \mathbf{Dom}(R_d, Y)$ $\rightarrow X \sqsubseteq Y$
38	ObjSom-ObjRng	$X \sqsubseteq \exists R_o.Y$ $\wedge\ \mathbf{Rng}(R_o, Z)$ $\rightarrow X \sqsubseteq \exists R_o.(Y \sqcap Z)$	$X \sqsubseteq\!\geq R_o.Y,\ n > 0$ $\wedge\ \mathbf{Rng}(R_o, Z)$ $\rightarrow X \sqsubseteq\!\geq nR_o.(Y \sqcap Z)$ $X \sqsubseteq\!= nR_o.Y,\ n > 0$ $\wedge\ \mathbf{Rng}(R_o, Z)$ $\rightarrow X \sqsubseteq\!= nR_o.(Y \sqcap Z)$
39	SubCls-SubCls-1	$X \sqsubseteq Y$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	$X \sqsubseteq Y$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$
40	SubCls-SubCls-2	$X \sqsubseteq Y$ $\wedge\ X \sqsubseteq Z$ $\rightarrow X \sqsubseteq (Y \sqcap Z)$	
41	ObjSom-ObjMin	$X \sqsubseteq \exists R_o.Y$ $\wedge\ \geq 1R_o.Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	$X \sqsubseteq\!\geq nR_o.Y,\ n > 0$ $\wedge\ \exists R_o.Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$ $X \sqsubseteq\!= nR_o.Y,\ n > 0$ $\wedge\ \exists R_o.Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$
42	DatSom-DatMin	$X \sqsubseteq \exists R_d.D_r$ $\wedge\ \geq 1R_d.D_r \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	$X \sqsubseteq\!\geq nR_d.D_r,\ n > 0$ $\wedge\ \exists R_d.D_r \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$ $X \sqsubseteq\!= nR_d.D_r,\ n > 0$ $\wedge\ \exists R_d.D_r \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$
43	ObjSom-SubCls	$X \sqsubseteq \exists R_o.Y$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq \exists R_o.Z$	$X \sqsubseteq\!\geq nR_o.Y,\ n \geq 0$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq\!\geq nR_o.Z$ $X \sqsubseteq\!= nR_o.Y,\ n \geq 0$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq\!= nR_o.Z$
44	ObjSom-SubObj	$X \sqsubseteq \exists R_o.Y$ $\wedge\ R_o \sqsubseteq S_o$ $\rightarrow X \sqsubseteq \exists S_o.Y$	$X \sqsubseteq\!\geq nR_o.Y,\ n \geq 0$ $\wedge\ R_o \sqsubseteq S_o$ $\rightarrow X \sqsubseteq\!\geq nS_o.Y$ $X \sqsubseteq\!= nR_o.Y,\ n \geq 0$ $\wedge\ R_o \sqsubseteq S_o$ $\rightarrow X \sqsubseteq\!= nS_o.Y$
45	ObjUni-SubCls	$X \sqsubseteq (Y \sqcup Z)$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	
46	ObjAll-ObjInv	$X \sqsubseteq \forall R_o.Y$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \exists S_o.X \sqsubseteq Y$	
47	ObjSom-ObjAll-1	$\exists R_o.Y \sqsubseteq X$ $\wedge\ \forall R_o.\bot \sqsubseteq X$	

Continued on Next Page...

			$\rightarrow \forall R_o.Y \sqsubseteq X$	
	49	ObjSom-ObjAll-2	$X \sqsubseteq \exists R_o.\top$ $\wedge\ X \sqsubseteq \forall R_o.Y$ $\rightarrow X \sqsubseteq \exists R_o.Y$	
	49	ObjSom-ObjTra	$X \sqsubseteq \exists R_o.(\exists R_o Y)$ $\wedge\ \mathbf{Tra}(R_o)$ $\rightarrow X \sqsubseteq \exists R_o.Y$	$X \sqsubseteq\!\geq nR_o.(\geq nR_oY)$, $n > 0$ $\wedge\ \mathbf{Tra}(R_o)$ $\rightarrow X \sqsubseteq\!\geq nR_o.Y$
	50	ObjDom-Bot	$\mathbf{Dom}(R_o, X)$ $\wedge\ X \sqsubseteq \bot$ $\rightarrow \top \sqsubseteq \forall R_o.\bot$	
	51	ObjRng-Bot	$\mathbf{Rng}(R_o, X)$ $\wedge\ X \sqsubseteq \bot$ $\rightarrow \top \sqsubseteq \forall R_o.\bot$	
3	52	DisCls-SubCls-SubCls	$\mathbf{Dis}(\mathbf{X}, \mathbf{Y})$ $\wedge\ U \sqsubseteq X$ $\wedge\ V \sqsubseteq Y$ $\rightarrow \mathbf{Dis}(U, V)$	
	53	SubCls-SubCls-DisCls	$X \sqsubseteq Y$ $\wedge\ X \sqsubseteq Z$ $\wedge\ \mathbf{Dis}(Y, Z)$ $\rightarrow X \sqsubseteq \bot$	
	54	ObjUni-SubCls-SubCls	$X \sqsubseteq (U \sqcup V)$ $\wedge\ U \sqsubseteq Z$ $\wedge\ V \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	
	55	ObjSom-ObjSom-ObjTra	$X \sqsubseteq \exists R_o.Y$ $\wedge\ Y \sqsubseteq \exists R_o.Z$ $\wedge\ \mathbf{Tra}(R_o)$ $\rightarrow X \sqsubseteq \exists R_o.Z$	$X \sqsubseteq\!\geq nR_o.Y$, $n > 0$ $\wedge\ Y \sqsubseteq\!\geq nR_o.Z$ $\wedge\ \mathbf{Tra}(R_o)$ $\rightarrow X \sqsubseteq\!\geq nR_o.Z$
	56	DatVal-DatVal-DatFun	$X \sqsubseteq \exists R_d.\{l_0 \star D_{t0}\}$ $\wedge\ X \sqsubseteq \exists R_d.\{l_1 \star D_{t1}\}$, D_{t0} & D_{t1} are disjoint, or $l_0 \neq l_1$ $\wedge\ \mathbf{Fun}(\mathbf{R_d})$ $\rightarrow X \sqsubseteq \bot$	
4	57	ObjVal-ObjVal-DifInd-ObjFun	$X \sqsubseteq \exists R_o.\{i\}$ $\wedge\ X \sqsubseteq \exists R_o.\{j\}$ $\wedge\ \mathbf{Dif}(\mathbf{i}, \mathbf{j})$ $\wedge\ \mathbf{Fun}(\mathbf{R_o})$ $\rightarrow X \sqsubseteq \bot$	

chain of similar or related rules will be investigated in order to make explanations more concise, and important inference steps become more visible to end-users.

Chapter 5

Construction of Proof Trees

This chapter describes the algorithms to construct *proof trees* from an entailment-justification pair based on the deduction rules described in Chapter 4. For our purposes, a proof tree is defined as any tree linking axioms of a justification (terminal nodes) to an entailment (root node), in such a way that every local tree (i.e., every non-terminal node and its children) corresponds to a deduction rule. This means that if the entailment and the justification already correspond to a deduction rule, no further non-terminal nodes (i.e., lemmas) need to be added. Otherwise, a proof tree needs to be sought. Figure 1.1 shows an example proof tree generated by our system for the entailment-justification pair in Table 1.1.

Unlike the construction of justification oriented proofs which is based on the computation of the deductive closure of an axiom set [Hor11], the construction of our proof trees is through exhaustive search of possible applications of the deduction rules. The main algorithm **ComputeProofTrees($\eta, \mathcal{J}$)** to compute proof trees is summarised in Algorithm 1, where $(\eta, \mathcal{J})$ is an input entailment-justification pair. In essence, the algorithm consists of two phases: first, superfluous parts in the justification, if present, are eliminated resulting in one or more *initial* trees, then for each initial tree, all *complete* proof trees are constructed. Details of these algorithms are described in the next two sections.

Algorithm 1 ComputeProofTrees($\mathcal{J}$, η)

1: $ProofList_{initial} \leftarrow$ ComputeInitialTrees($\mathcal{J}$, η)
2: **for** $P_{initial} \in ProofList_{initial}$ **do**
3: $\quad ProofList_{complete} \leftarrow$ ComputeCompleteProofTrees($P_{initial}$)
4: $\quad ProofList_{result}.addAll(ProofList_{complete})$
5: **end for**
6: **return** $ProofList_{result}$

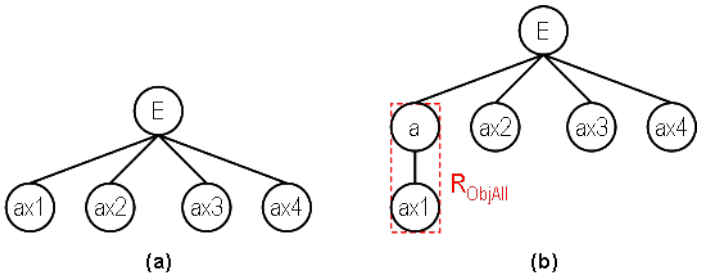


Figure 5.1: Examples of initial trees for two types of justifications: (a) justifications without superfluous parts, and (b) justifications in which axiom 1 contains unnecessary parts for the entailment. The label 'R_{ObjAll}' refers to rule 'ObjAll' listed in Table 4.3.

5.1 Computing Initial Trees

5.1.1 Method

For our purpose, an initial tree is as an incomplete proof tree in which the root node is an entailment, the terminal nodes are axioms of a justification, and for each terminal node that contains unnecessary parts, a non-terminal node that links the root node to the associated terminal node is added. Such a non-terminal node, if existing, and the associated terminal nodes form a local tree which corresponds to a single-premise deduction rule (described in Section 4.4). This means each non-terminal node is a lemma that unpacks part of the meaning of the associated axiom in the justification—the part that actually contributes to the entailment. If none of the axioms in the justification contains superfluous parts, then no non-terminal nodes will be added into the initial tree. As an example, Figure 5.1 shows two initial trees for two types of justifications, one with and one without unnecessary parts. Case (b) is the initial tree for the proof tree in Figure 1.1.

To generate initial trees for an arbitrary entailment-justification pair, axioms in the justification that have superfluous parts need to be identified first. This identification requires analysis of the *contribution* of each axiom to the entailment, but *not* analysis of the structure and semantics of axioms individually. This is because given an OWL axiom, many parts of its meaning can be unpacked, and the more complex the axiom, the more parts

can be unpacked. For instance, from the axiom $A \equiv B \sqcap C \sqcap D$, any of the following parts of its meaning can be unpacked: $A \sqsubseteq B$, $A \sqsubseteq C$, $A \sqsubseteq D$, $A \sqsubseteq B \sqcap C$, $A \sqsubseteq B \sqcap D$, $A \sqsubseteq C \sqcap D$, and $A \sqsubseteq B \sqcap C \sqcap D$. Without the analysis of the entailment-justification pair as a whole, each of these parts needs to be tested to find out the appropriate ones. For large justifications, the number of such tests increases significantly, and will cause a large computational overhead.

To avoid the above-mentioned overhead, Horridge et al.'s algorithm [HPS10, Hor11] is employed to compute laconic justifications. In particular, given an entailment-justification pair, Horridge et al.'s algorithm is used to compute all laconic justifications for the entailment from the original justification (the original justification is treated as an ontology of the entailment). As explained in Section 2.3.3, this algorithm returns one or more laconic justifications for the entailment. If axioms in the original justification contain no superfluous parts, the original justification will be returned as a unique laconic justification. Thereafter, an initial proof tree is constructed from each laconic justification.

To construct an initial tree, a tree in which the root node is the entailment, and all terminal nodes are axioms in the original justification is initialised. For each axiom in the laconic justification, the Pellet reasoner [SPG$^+$07] is queried to identify from which axiom in the original justification this axiom follows. If the two axioms are identical, no lemma will be added. Otherwise, the axiom in the laconic justification will be used as a lemma, and a non-terminal node containing this lemma will be added between the root node and the associated terminal node. The result of this process is a number of initial trees, one from each laconic justification.

5.1.2 Exceptions

On close inspection we have found that in a number of cases, the elimination of superfluous parts according to Horridge et al.'s algorithm [HPS10, Hor11] is not appropriate for explanation purposes because it produces trivial inference steps. For example, given a justification including two premises (1) $PetOwner \sqsubseteq \exists hasPet.Pet$ (Every pet owner has a pet) and (2) $\mathbf{Dom}(hasPet, Person)\}$ (Anything that has as pet something is a person), and an entailment $PetOwner \sqsubseteq Person$ (Every pet owner is a person), Horridge et al.'s algorithm produces the following laconic justification: (1) $PetOwner \sqsubseteq \exists hasPet.\top$

(Every pet owner has as pet something) and (2) $\mathbf{Dom}(hasPet, Person)\}$ (Anything that has a pet is a person). In the laconic justification, the filler *Pet* in axiom 1 is omitted as it is not required for the entailment. According to our algorithm, an inference step which unpacks $PetOwner \sqsubseteq \exists hasPet.\top$ (Every pet owner has as pet something) from $PetOwner \sqsubseteq \exists hasPet.Pet$ (Every pet owner has a pet) will be added to the resulting initial tree. However, we believe that this inference step is trivial and should be ignored. Therefore, such an elimination should be skipped. We call such cases *exceptions*.

The rest of this sub-section describes all of the exceptions we manually collected through the analysis of our corpus, and how we dealt with them. In this description, C, C_k, D, and D_k are class names or expressions, R_o and R_{ok} are object property names, R_d and R_{dk} are data property names, i and i_k are individual names, l and l_k are literal values, where $k \geq 1$.

Case 1: Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \sqsubseteq \exists R_o.D$

As explained in the above example, this unpacking is ignored in order to avoid adding the trivial step which yields $C \sqsubseteq \exists R_o.\top$ from $C \sqsubseteq \exists R_o.D$ into the initial tree. This means no single-premise rule is required here. This strategy is also applied to a more general case in which $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\top)$ is unpacked from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.D_p)$, where $M \geq 1$ and $N \geq 1$.

A similar strategy is also applied to the following cases:

- Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \equiv \exists R_o.D$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.D_p)$, where $M \geq 1$ and $N \geq 1$; this results in an inference step corresponding to rule 'EquCls' (to yield $C \sqsubseteq \exists R_o.D$ from $C \equiv \exists R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \sqsubseteq C_1 \sqcap \exists R_o.D$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \exists R_{op}.\top)$ from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.D_p)$, where $M \geq L \geq 1$, $N \geq K \geq 1$, and $M+N > L+K$; this results in an inference step corresponding to rule 'ObjInt-2' (to yield $C \sqsubseteq \exists R_o.D$ from $C \sqsubseteq C_1 \sqcap \exists R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \equiv C_1 \sqcap \exists R_o.D$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \exists R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.D_p)$, where $M \geq L \geq 1$, $N \geq K \geq 1$, and $M+N > L+K$; this results in an inference step corresponding to rule 'ObjInt-1'

(to yield $C \sqsubseteq \exists R_o.D$ from $C \equiv C_1 \sqcap \exists R_o.D$) in the initial tree.

Case 2: Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \sqsubseteq \exists R_o.\{i\}$

In this case, the filler $\{i\}$ is omitted because it is not necessary for the entailment. This omission causes the trivial step which yields $C \sqsubseteq \exists R_o.\top$ from $C \sqsubseteq \exists R_o.\{i\}$ to be added into the initial tree. To avoid this, the filler $\{i\}$ is retained, so no single-premise rule is required here. This strategy is also applied to a more general case in which $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\top)$ is unpacked from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\{i_p\})$, where $M \geq 1$ and $N \geq 1$.

A similar strategy is also applied to the following cases:

- Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \equiv \exists R_o.\{i\}$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\{i_p\})$, where $M \geq 1$ and $N \geq 1$; this results in an inference step corresponding to rule 'EquCls' (to yield $C \sqsubseteq \exists R_o.\{i\}$ from $C \equiv \exists R_o.\{i\}$) in the initial tree.

- Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \sqsubseteq C_1 \sqcap \exists R_o.\{i\}$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \exists R_{op}.\top)$ from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\{i_p\})$, where $M \geq L \geq 1$, $N \geq K \geq 1$, and $M + N > L + K$; this results in an inference step corresponding to rule 'ObjInt-2' (to yield $C \sqsubseteq \exists R_o.\{i\}$ from $C \sqsubseteq C_1 \sqcap \exists R_o.\{i\}$) in the initial tree.

- Unpacking $C \sqsubseteq \exists R_o.\top$ from $C \equiv C_1 \sqcap \exists R_o.\{i\}$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \exists R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{op}.\{i_p\})$, where $M \geq L \geq 1$, $N \geq K \geq 1$, and $M + N > L + K$; this results in an inference step corresponding to rule 'ObjInt-1' (to yield $C \sqsubseteq \exists R_o.\{i\}$ from $C \equiv C_1 \sqcap \exists R_o.\{i\}$) in the initial tree.

Case 3: Unpacking $C \sqsubseteq \geq nR_o.\top$ from $C \sqsubseteq \geq nR_o.D$, where $n \geq 0$

In this case, the filler D is retained to avoid adding the trivial step which yields $C \sqsubseteq \geq nR_o.\top$ from $C \sqsubseteq \geq nR_o.D$ to the initial tree. This means no single-premise rule is required here. This strategy is also applied to a more general case in which $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \geq n_p R_{op}.\top)$ is unpacked from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \geq n_p R_{op}.D_p)$, where $M \geq 1$, $N \geq 1$, and $n_p \geq 0$.

A similar strategy is also applied to the following cases:

- Unpacking $C \sqsubseteq \geq n_2 R_o.\top$ from $C \sqsubseteq = n_1 R_o.D$, where $n_1 \geq n_2 \geq 0$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \geq n_{2p} R_{op}.\top)$ from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_{1p} R_{op}.D_p)$, where $M \geq 1$, $N \geq 1$, and $n_{1p} \geq n_{2p} \geq 0$; this results in an inference step corresponding to rule 'ObjExt' (to yield $C \sqsubseteq \geq n_2 R_o.D$ from $C \sqsubseteq = n_1 R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \geq n_2 R_o.\top$ from $C \equiv = n_1 R_o.D$, where $n_1 \geq n_2 \geq 0$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \geq n_{2p} R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_{1p} R_{op}.D_p)$, where $M \geq 1$, $N \geq 1$, and $n_{1p} \geq n_{2p} \geq 0$; this results in two inference steps corresponding to rules 'EquCls' (to yield $C \sqsubseteq = n_1 R_o.D$ from $C \equiv = n_1 R_o.D$) and 'ObjExt' (to yield $C \sqsubseteq \geq n_2 R_o.D$ from $C \sqsubseteq = n_1 R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \geq n_2 R_o.\top$ from $C \sqsubseteq C_1 \sqcap = n_1 R_o.D$, where $n_1 \geq n_2 \geq 0$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \geq n_{2p} R_{op}.\top)$ from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_{1p} R_{op}.D_p)$, where $M \geq L \geq 1$, $N \geq K \geq 1$, $M + N > L + K$, and $n_{1p} \geq n_{2p} \geq 0$; this results in two inference steps corresponding to rules 'ObjInt-2' (to yield $C \sqsubseteq = n_1 R_o.D$ from $C \sqsubseteq C_1 \sqcap = n_1 R_o.D$) and 'ObjExt' (to yield $C \sqsubseteq \geq n_2 R_o.D$ from $C \sqsubseteq = n_1 R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \geq n_2 R_o.\top$ from $C \equiv C_1 \sqcap = n_1 R_o.D$, where $n_1 \geq n_2 \geq 0$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \geq n_{2p} R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_{1p} R_{op}.D_p)$, where $M \geq L \geq 1$, $N \geq K \geq 1$, $M + N > L + K$, and $n_{1p} \geq n_{2p} \geq 0$; this results in two inference steps corresponding to rules 'ObjInt-1' (to yield $C \sqsubseteq = n_1 R_o.D$ from $C \equiv C_1 \sqcap = n_1 R_o.D$) and 'ObjExt' (to yield $C \sqsubseteq \geq n_2 R_o.D$ from $C \sqsubseteq = n_1 R_o.D$) in the initial tree.

Case 4: Unpacking $C \sqsubseteq \leq n R_o.\top$ from $C \sqsubseteq \leq n R_o.D$, where $n \geq 0$

In this case, the filler D is retained to avoid adding the trivial step which yields $C \sqsubseteq \leq n R_o.\top$ from $C \sqsubseteq \leq n R_o.D$ to the initial tree. This means no single-premise rule is required here. This strategy is also applied to a more general case in which $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \leq n_p R_{op}.\top)$ is unpacked from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \leq n_p R_{op}.D_p)$, where $M \geq 1$, $N \geq 1$, and $n_p \geq 0$.

A similar strategy is also applied to the following cases:

- Unpacking $C \sqsubseteq \leq n R_o.\top$ from $C \sqsubseteq = n R_o.D$, where $n \geq 0$, and more generally,

$C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \leq n_p R_{op}.\top)$ from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_p R_{op}.D_p)$, where $M \geq 1$, $N \geq 1$, and $n_p \geq 0$; this results in one inference step corresponding to rule 'ObjExt' (to yield $C \sqsubseteq \leq n R_o.D$ from $C \sqsubseteq = n R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \leq n R_o.\top$ from $C \equiv = n R_o.D$, where $n \geq 0$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \leq n_p R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_p R_{op}.D_p)$, where $M \geq 1$, $N \geq 1$, and $n_p \geq 0$; this results in two inference steps corresponding to rules 'EquCls' (to yield $C \sqsubseteq = n R_o.D$ from $C \equiv = n R_o.D$) and 'ObjExt' (to yield $C \sqsubseteq \leq n R_o.D$ from $C \sqsubseteq = n R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \leq n R_o.\top$ from $C \sqsubseteq C_1 \sqcap = n R_o.D$, where $n \geq 0$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \leq n_p R_{op}.\top)$ from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_p R_{op}.D_p)$, where $M \geq L \geq 1$, $N \geq K \geq 1$, $M + N > L + K$, and $n_p \geq 0$; this results in two inference steps corresponding to rules 'ObjInt-2' (to yield $C \sqsubseteq = n R_o.D$ from $C \sqsubseteq C_1 \sqcap = n R_o.D$) and 'ObjExt' (to yield $C \sqsubseteq \leq n R_o.D$ from $C \sqsubseteq = n R_o.D$) in the initial tree.

- Unpacking $C \sqsubseteq \leq n R_o.\top$ from $C \equiv C_1 \sqcap = n R_o.D$, where $n \geq 0$, and more generally, $C \sqsubseteq (\sqcap_{k=1}^{L} C_k) \sqcap (\sqcap_{p=1}^{K} \leq n_p R_{op}.\top)$ from $C \equiv (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} = n_p R_{op}.D_p)$, where $M \geq L \geq 1$, $N \geq K \geq 1$, $M + N > L + K$, and $n_p \geq 0$; this results in two inference steps corresponding to rules 'ObjInt-1' (to yield $C \sqsubseteq = n R_o.D$ from $C \equiv C_1 \sqcap = n R_o.D$) and 'ObjExt' (to yield $C \sqsubseteq \leq n R_o.D$ from $C \sqsubseteq = n R_o.D$) in the initial tree.

Case 5: Unpacking $C \sqsubseteq \exists R_d.Literal$ from $C \sqsubseteq \exists R_d.\{l\} \star D_t$

In this case, the filler $\{l\} \star D_t$ is omitted because it is not necessary for the entailment. This omission causes the trivial step which yields $C \sqsubseteq \exists R_d.Literal$ from $C \sqsubseteq \exists R_d.\{l\} \star D_t$ to be added to the initial tree. To avoid this, the filler $\{l\} \star D_t$ is retained, so no single-premise rule is required here. This strategy is also applied to a more general case in which $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{dp}.Literal)$ is unpacked from $C \sqsubseteq (\sqcap_{k=1}^{M} C_k) \sqcap (\sqcap_{p=1}^{N} \exists R_{dp}.\{l_p\} \star D_{tp})$, where $M \geq 1$ and $N \geq 1$.

Case 5: Unpacking $R_o \sqsubseteq \mathbf{Inv}(S_o)$ from $\mathbf{Invs}(R_o, S_o)$

Horridge et al.'s algorithm [HPS08b] always transforms the axiom $\mathbf{Invs}(R_o, S_o)$ to $R_o \sqsubseteq \mathbf{Inv}(S_o)$. This causes the trivial inference step which unpacks $R_o \sqsubseteq \mathbf{Inv}(S_o)$ from $\mathbf{Invs}(R_o, S_o)$

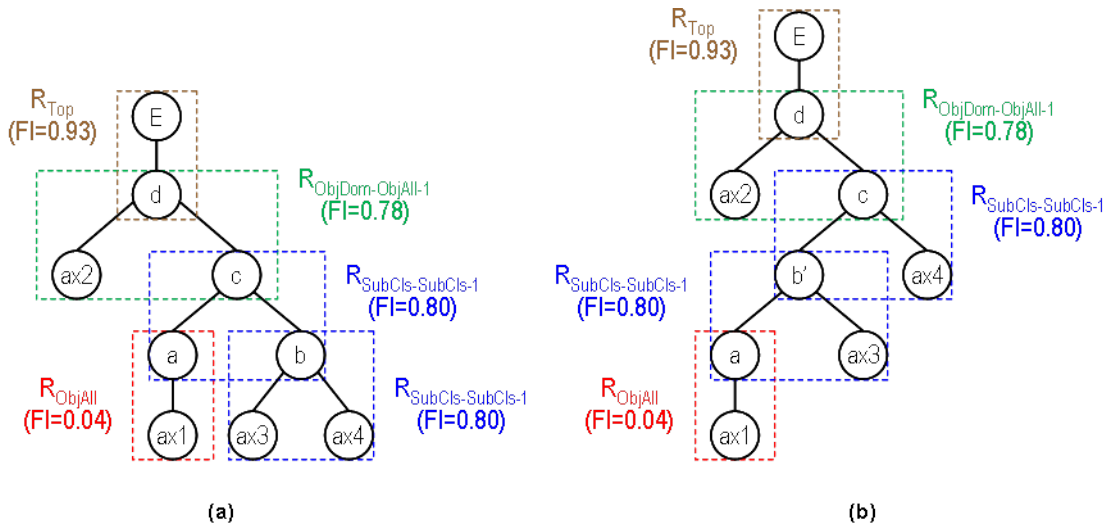


Figure 5.2: Two complete proof trees generated for the entailment-justification pair in Table 1.1; they are constructed from the initial tree (b) in Figure 5.1. In this algorithm, 'E' means the entailment.

to be added to the initial tree. To avoid this, the original axiom $\mathbf{Invs}(R_o, S_o)$ is retained, so no single-premise rule is required in this case.

5.2 Computing Proof Trees

Complete proof trees are constructed from initial trees. For a given initial tree, a proof tree can be sought by searching for all applications of the deduction rules, where possible, on the *immediate* child nodes (either terminal or non-terminal) of the root node, so introducing lemmas and growing the tree bottom-up towards the root node. Exhaustive search using this method may yield zero (if the initial tree is already complete), one, or multiple proof trees. In other words, there might be zero, one, or multiple proof trees for a given entailment-justification pair. For example, from the entailment-justification pair in Table 1.1, two proof trees can be generated from the basis of the initial tree (b) in Figure 5.1, and they are shown in Figure 5.2.

The algorithm for computing complete proof trees is summarised in Algorithm 2. It uses two sub-routines. The first sub-routine **ComputeAllPartitions(***NodeList***)** is an exhaustive search algorithm which computes all possible ways to partition a node list so that at least one deduction rule is applied in each partition. For instance, given a list of four nodes $[N_1, N_2, N_3, N_4]$, a possible partition of this node list is $\langle (N_1, N_3, R_a), (N_2, R_b), N_4 \rangle$ in which N_1 and N_3 can be combined to give rise to a conclusion based on rule R_a, and

N_2 can be transformed into a new axiom through rule R_b. Another partition of this node list is $\langle (N_1, N_3, R_a), (N_2, N_4, R_c) \rangle$ in which, in addition to the combination of N_1 and N_3, N_2 and N_4 can be combined to give rise a conclusion based on R_c. In such partitions, the identification of the rule associated with each group, where possible, requires a search through the rule set. In order to improve the feasibility of this search, factors related to the logical structure of rules such the number of premises, the types of premises, the number, types, and distribution of arguments etc. are used to filter inappropriate rules.

Algorithm 2 ComputeCompleteProofTrees($P_{initial}$)

```
1: ProofList_complete ← {}
2: ProofList_incomplete ← {P_initial}
3: ProofList'_incomplete
4: while !ProofList_incomplete.empty do
5:     for P_incomplete ∈ ProofList_incomplete do
6:         E ← P_incomplete.root
7:         NodeList ← P_incomplete.root.children
8:         ProofList'_incomplete ← {}
9:         if NodeList = {A} and A → E conforms to rule R_1 then
10:            P'_incomplete ← A copy of P_incomplete
11:            ProofList_complete.add(P'_incomplete)
12:        else
13:            PartitionList ← ComputeAllPartitions(NodeList)
14:            for Partition ∈ PartitionList do
15:                P_new ← ComputeProofByApplyingPartition(P_incomplete, Partition)
16:                if P_new! = null and P_new = P_incomplete then
17:                    ProofList_complete.add(P_incomplete)
18:                else if P_new! = null then
19:                    ProofList'_incomplete.add(P_new)
20:                end if
21:            end for
22:        end if
23:    end for
24:    ProofList_incomplete ← ProofList'_incomplete
25: end while
```

The second sub-routine **ComputeProofByApplyingPartition(**$P_{incomplete}$**,** $Partition$**)** is to compute a new proof tree after applying a partition on the input proof tree. As in the main algorithm, this application is based on the *immediate* child nodes of the root node. If these child nodes and the root node already conform to a rule, this means the proof tree is already complete, so will be updated (with the information of the rule) and returned. Otherwise, the rules within the partition will be applied on a copy of the input proof tree, and new lemma nodes will be inserted into the tree. The new proof tree will be returned as a new *incomplete* proof tree. If any error occurs in this sub-routine, a null proof tree will be returned.

Table 5.1: Results of the feasibility test

Entailment Type	**Passed Cases**	**Failed Cases by**		**TOTAL**
		Rule Coverage	**Time Out**	
$\top \sqsubseteq A$	24 (85.7%)	4 (14.3%)	0 (0.0%)	28 (100%)
$A \sqsubseteq \bot$	1,799 (93.2%)	123 (6.4%)	8 (0.4%)	1,930 (100%)
$A \sqsubseteq B$	114,495 (75.4%)	32,954 (21.7%)	4,401 (2.9%)	151,850 (100%)
TOTAL	116,318 (75.6%)	37,490 (21.5%)	4,409 (2.9 %)	153,808 (100%)

5.3 Feasibility of Computing Proof Trees

The exhaustive search algorithm used in the sub-routine **ComputeAllPartitions(**$NodeList$**)** is a computational limitation of our algorithm, especially for large justifications. The higher the justification size, the higher the computational overhead, and so the longer time for the computation. To assess the feasibility of our algorithm, we employed it to construct proof trees for entailment-justification pairs collected from our ontology corpus (described in Chapter 4). We restricted to only justifications of size ten or less, and relied on the rule set listed in Table 4.3 to construct proof trees. The time limitation for computing a single proof tree was set to 60 seconds. The test was deployed and run on a super-computer with a 64-bit Java virtual machine installed. The purpose of using the super-computer for this test was to maximise the number of entailment-justification pairs computed from our ontology corpus.

The result of this test is summarised in Table 5.1. The test showed that our algorithm was robust enough to generate proof trees, and so explanations, for 75.6% (or 116,318) of the input justification-entailment pairs. Among over 37,000 cases failed, over 97 percent of them were caused by the limited coverage of the rule set; only 3 percent of them were caused by the time out set-up (mainly caused by the exhaustive search algorithm used in the sub-routine **ComputeAllPartitions(**$NodeList$**)**), and the justification size of these cases was 10. These results have confirmed that our algorithm performs relatively well with justifications having ten or fewer axioms. Larger justifications are too difficult to be understood and used by human developers in practice, so it is fair to say that our algorithm works well for explanation purpose.

5.4 Conclusions and Future Work

We have proposed an efficient algorithm to construct proof trees from an entailment-justification pair based on a set of deduction rules. We have also performed an extensive test of feasibility of the algorithm (over 150,000 entailment-justification pairs from 171 ontologies), and shown that it is robust enough to construct proof trees, and so explanations, for most cases.

Improving the feasibility of the algorithm on large justifications is not a focus of our research, and we would be happy for others to plug in better solutions here. Constructing proof trees at a higher level of abstraction (based on abstract deduction rules) is beyond the scope of this thesis, but would be investigated as part of future work.

Chapter 6

Verbalisations for Deduction Rules

This chapter describes the identification of verbalisations for the premises and entailment of our deduction rules. Section 6.1 discusses the sources of variability, and factors that affect the number of verbalisations for a rule. A rule can only be verbalised clearly if its constituent axioms are verbalised clearly—a task addressed by much existing NLG research. Hence, in Section 6.2, the state-of-the-art of the ontology verbalisation task is discussed, followed by the description of our selection of verbalisations for basic axioms in OWL. Subsequently, the selection of verbalisations for most of our deduction rules is described in Section 6.3. Finally, Section 6.4 presents an empirical study constructed to identify the best verbalisations for rules that have multiple candidate verbalisations.

6.1 Diversity of Verbalisations for Deduction Rules

Given a deduction rule, there are always multiple ways to translate it into an English text, and the number of possible verbalisations depends on several factors. Firstly, it depends on the number of *ways to verbalise the premises and entailment* within the rule. Secondly, it depends on the *number of premises*. This is because each verbalisation is associated with

Table 6.1: All possible verbalisations for the premises and entailment of the deduction rule 'SubCls-DisCls' in Table 4.3, which unpacks $X \sqsubseteq \bot$ from two premises (1) $X \sqsubseteq Y$ and (2) $\mathbf{Dis}(X, Y)$

ID	Premises	Entailment
1	Every X is a Y. No X is a Y.	Nothing is a X.
2	Every X is a Y. Nothing is both a X and a Y.	Nothing is a X.
3	Every X is a Y. No Y is a X.	Nothing is a X.
4	Every X is a Y. Nothing is both a Y and a X.	Nothing is a X.
5	No X is a Y. Every X is a Y.	Nothing is a X.
6	Nothing is both a X and a Y. Every X is a Y.	Nothing is a X.
7	No Y is a X. Every X is a Y.	Nothing is a X.
8	Nothing is both a Y and a X. Every X is a Y.	Nothing is a X.

an ordered sequence of premises; hence, the more premises the more possible orderings (in the worst case, there are $n!$ orderings for n premises). As a consequence of this analysis, any rules with more than one premise would have multiple verbalisations. The last but not least factor is the appearance of *commutative OWL constructors* such as **Dis** and $\sqcap$. This is because each verbalisation is also associated with an ordered sequence of arguments in axioms. Therefore, "No cat is a dog" and "No dog is a cat", for instance, are two different verbalisations. The more commutative constructors a deduction rule has, the higher number of verbalisations.

As an example, let us work out some possible verbalisations of the premises and entailment of rule 'SubCls-DisCls' in Table 4.3, which unpacks $X \sqsubseteq \bot$ from two premises: (1) $X \sqsubseteq Y$ and (2) $\mathbf{Dis}(X, Y)$. Assuming that only the following two verbalisations for the axiom $\mathbf{Dis}(X, Y)$ are considered here: (1) "No X is a Y" and (2) "Nothing is both an X and a Y". Because of the commutation of this axiom, two additional verbalisations need to be considered here: (3) "No Y is an X" and (4) "Nothing is both a Y and an X". Since this rule has two premises, there are two ways of ordering the premises. All of these factors lead to a total of 2*2*2 or 8 different verbalisations for the premises and entailment of the rule, as listed in Table 6.1.

According to the above analysis, all rules, even those having only one premise, may have multiple verbalisations. Moreover, the number of all possible verbalisations of a rule is often non-trivial. A rule with only two premises, such as the one in the example, may result in eight different verbalisations. Many of our rules consist of three or even four premises, leading to even higher numbers of verbalisations due to different premise orderings—e.g., three-premise rules create 6 premise orderings while four-premise rules create 24 premise orderings. Since some verbalisations may be easier to understand than others, this raises a new research problem of identifying the most understandable verbalisation among a set of alternatives for a rule.

6.2 Verbalisations for OWL Axioms

6.2.1 State of the Art

As mentioned in Chapter 1, the opacity of standard OWL syntaxes, such as RDF/XML [RDF04], is a source of difficulty in building OWL ontologies [ST04]. This problem was addressed through graphical tools which offer the use of tabs, trees, wizards, etc. for viewing and editing OWL ontologies—e.g., TopBraid Composer [Top], Protégé [KMR04], and Swoop [KPS+06]—and more user-friendly formats such as Functional OWL syntax [OWL12c] and Manchester OWL syntax [HDG+06, HPS08c].

Dzbor et al. [DMB+06] conducted a empirical study to compare TopBraid Composer and Protégé, and found that both OWL experts and non-experts experienced difficulties in getting acquainted with these tools, and were not satisfied with them. Functional OWL syntax [OWL12c] provides quasi-English descriptions of OWL constructors such as *ClassAssertion(Insectivore, keroppi)* ("Keroppi is an insectivore") and *ObjectPropertyDomain(hasTopping, Pizza)* ("Anything that has a topping is a pizza"), which are often unnatural. However, this syntax helps to show the formal structure of OWL axioms and expressions clearly, and hence is preferable in some cases.

In Manchester OWL syntax [HDG+06, HPS08c], all information about a particular class, property, or individual is collected into a single construct (called a *frame*), and all OWL constructors are replaced by intuitive English glosses—e.g., axiom constructors *EquivalentClasses*, *DisjointClasses*, and *ObjectPropertyDomain* are replaced by 'EquivalentTo',

'DisjointWith', and 'Domain'; class constructors *ObjectIntersectionOf*, *ObjectUnionOf*, *ObjectSomeValuesFrom*, and *ObjectAllValuesFrom* are replaced by 'and', 'or', 'some', and 'only'—thus it is more concise, easier to read and write than other syntaxes. Although the heavy use of parentheses in this syntax suggests the need of some knowledge of OWL, evaluation studies conducted by Horridge et al. [HDG+06] suggest that it has been well received by non-logicians. Therefore, it has been used as the default OWL syntax in Protégé [KMR04], and supported in TopBraid Composer [Top].

In addition to the syntax, the lack of understanding of OWL's underlying logic-based semantics is another source of difficulty in building OWL ontologies. Rector et al. [RDH+04] identified a list of common logical issues that new users often encounter when they develop OWL ontologies by using the Manchester OWL syntax, including the followings:

1. Trivial satisfiability of universal restrictions—that 'only' ($\forall$) does not imply 'some' ($\exists$) because $\forall R_o.C$ subsumes $\forall R_o.\bot$
2. Confusion about the representation of "some not" (i.e., $\exists R_o.(\neg C)$) and "not some" (i.e., $\neg(\exists R_o.C)$)
3. The difference between the linguistic and logical usage of 'and' and 'or'
4. The effect of range and domain constraints
5. The difference between defined and primitive classes, and the mechanics of converting one to the other
6. Confusion about OWL's *open world* reasoning assumption—i.e., something is false only if it can be proved to contradict other information specified in the ontology
7. Mistaken use of $\forall$ rather than existential restrictions $\exists$ as the default restriction

The main reason for those issues is because the English glosses in Manchester OWL syntax only help to reveal general aspects of the semantics of axioms (but not their complete semantics), and hence are not enough to help inexperienced users. Rector and colleagues [RDH+04], based on their teaching experience through a series of workshops, tutorials, and teaching modules, suggested that *paraphrasing* OWL axioms can help users understand their precise meanings, and so avoid the above-mentioned issues. They also suggested a list of English paraphrases for basic OWL constructors. These suggestions

motivated a number of later efforts in providing more natural representations of axioms in OWL ontologies—this task is called *ontology verbalisation*.

Hewlett et al. [HKKHW05] developed an algorithm to verbalise class expressions in OWL into English text. To improve the readability of the text, a part-of-speech tagger was used to analyse the structure of property names in order to identify appropriate English paraphrases for them. Halaschek-Wiener et al. [HWGP+06] then extended Hewlett et al.'s algorithm to handle individuals. They also conducted an evaluation study which confirmed that natural language representations of OWL class descriptions were preferred by the participants to mathematical notation, Turtle (Terse RDF Triple Language) [BBLPC], OWL Abstract Syntax [OWL04a], and RDF/XML [RDF04], and this preference was statistically reliable. Jarrar et al. [JKD06] proposed an approach of verbalising an OWL ontology into multiple languages. In this approach, the structural templates of OWL statements were first identified, and verbalisations were generated from pre-defined templates in the associated language.

A common deficiency of the above-mentioned approaches is that they do not provide a basis for checking whether the resulting verbalisations are *unambiguous* for machine interpretability. As a consequence, the resulting verbalisations become a dead-end, and OWL statements cannot be reproduced from text. To avoid the risk of this ambiguity, Kaljurand and Fuchs [KF07] suggested a new approach based on *controlled natural languages* in order to enable round-trip ontology authoring [DIF+08].

Generally speaking, a controlled natural language (CNL) (sometimes simply called a "controlled language") is an engineered subset of a natural language (usually of English) created by restricting the grammar, lexicon, and style of the full natural language in order to reduce or eliminate ambiguity and complexity of the full natural language [Kit03]. Traditionally, CNLs are classified into two categories, namely *human-oriented* and *machine oriented* [WO98]. Human-oriented CNLs are those designed for being used in technical documents in order improve human comprehension. A prominent CNL of this type is AECMA[1] Simplified English [AEC04], designed to make aircraft support and maintenance procedures easier to understand by non-native speakers. Machine-oriented CNLs, on the other hand, are those designed to improve the text consumption by computer programs. A typical CNL of this type is KANT Controlled English [Mit99], designed to

[1] AECMA is the French acronym for the European Association of Aerospace Industries.

improve the automatic translation of technical documents to other languages.[2]

In knowledge representation, a number of machine-oriented CNLs have been proposed to provide a means for viewing and editing OWL ontologies in natural language. These CNLs are often underpinned by a formal logic-based semantics in order to enable backward mapping from text to OWL, and so allows users to encode an OWL ontology by entering knowledge in natural language text—this task is called *ontology authoring* as opposed to *ontology verbalisation*. Schwitter [Sch10] and Kuhn [Kuh10] defined four requirements for an ideal CNL for ontology authoring, including clearness (i.e., having a well-defined syntax and semantics), naturalness (i.e., producing sentences that are readable and unambiguously understandable for humans), simplicity (i.e., can be efficiently interpreted by computer programs and easily comprehended by humans), and expressivity (i.e., fully covering the desired problem domain). These requirements may nevertheless be conflict in practice, for example, between naturalness and expressivity [Pow12].

Notable examples of CNLs for authoring OWL ontologies are ACE (Attempto Controlled English) [KF07], SOS (Sydney OWL Syntax) [CSM07], CLOnE (Controlled Language for Ontology Editing) [FTB+07], Rabbit [HJD08], and more recently OWL Simplified English [Pow12].[3] These CNLs share a common design principle of mapping an OWL axiom to an English sentence, and vice versa. They also agree on how to realise many OWL constructors [SKC+08], both for axioms and class expressions—e.g., they all verbalise the axiom $Loch \sqsubseteq locatedIn.\{scotland\}$ as "Every loch is located in Scotland"; they all use the phrases "exactly", "at least", and "at most" to verbalise the constructors '$=$', '$\geq$', and '$\leq$'. In other cases, the resulting sentences differ in style and/or the expression of certain OWL constructors and entity names—e.g., property names.

6.2.2 Selection of Verbalisations

As discussed previously, various ways to translate an OWL axiom into an English sentence have been proposed; notable examples are CNLs such as ACE [KF07], Rabbit [HJD08], SOS [CSM07], and OWL Simplified English [Pow12]. These CNLs, although different, agree on how to verbalise many OWL constructors. Rabbit and ACE verbalisations are empirically tested [HJD08, Kuh13]), and confirmed to be readable, comprehensible, and

[2] See Kuhn's latest article [Kuh13] for a complete discussion on the classification of CNLs.
[3] See Kuhn's latest article [Kuh13] for the full list of existing CNLs and their details.

preferred by most of the participants. SOS verbalisations are confirmed to share substantial commonality with those of Rabbit and ACE; most of the differences are in style and the expression of certain OWL constructors [SKC+08]. OWL Simplified English [Pow12] is designed to allow only verbalisations that are structurally unambiguous sentences in English.

Providing the above-mentioned advantages of the four CNLs, we selected verbalisations for OWL axioms from those produced by these languages. Specifically, for each basic axiom in OWL, we collected four verbalisations produced by these languages, then selected the one that we believed to be the most concise and understandable. Many of the selected verbalisations were those provided by OWL Simplified English [Pow12]. Table 6.2 lists the selected verbalisations for basic OWL axioms. The original CNLs of these verbalisations are also presented in the table.

For most cases, only one verbalisation was selected for each axiom. However, two different verbalisations were selected for the axioms $A \equiv B$ and also $\mathbf{Dis}(A, B)$. This was because it was unclear for us which verbalisation was better. The second verbalisation for $\mathbf{Dis}(A, B)$ ("Nothing is both an A and a B") was not collected from the CNLs but formulated by ourselves. To identify the best verbalisation in these cases, we conducted an empirical study to identify which one was more understandable to the participants. This study will be described in detail in the subsequent sections of this chapter.

6.3 Verbalisations of Deduction Rules

This section addresses the problem of identifying the most understandable verbalisation (among a set of alternatives) for a deduction rule. The best solution for this problem is to test all candidate verbalisations of a rule empirically to identify which one is understood best by human reasoners. However, the number of rules is non-trivial, and, as explained previously, the number of all possible verbalisations for a rule is also non-trivial—due to the practical limitation of this PhD project, only a limited number of cases can be tested. Therefore, we based on existing theoretical insights from the psychology of reasoning to limit the number of test cases. In particular, we found that not all verbalisations of a rule are good for understanding the relevant inference as they present the premises and/or arguments in a conflicting order, and such verbalisations can be ruled out by relying on

Table 6.2: Verbalisations of OWL axioms with 'S' means the source CNL(s) which produces the verbalisation, 'C' means whether the associated axiom is commutative, 'OSE' stands for "OWL Simplified English"; A and B are class names, R_o and S_o are object property names, R_d and S_d are data property names, D_r is a data range, D_t is a data type, and a and b are individual names

OWL Axiom	**Verbalisation**	**S**	**C**
$A \sqsubseteq B$	Every A is a B.	OSE, ACE, SOS	
$A \equiv B$	1. An A is anything that is a B. 2. Every A is a B; every B is an A.	OSE ACE	✓
$\mathbf{Dis}(A, B)$	1. No A is a B. 2. Nothing is both an A and a B.	OSE, ACE	✓
$\mathbf{Dom}(R_o, A)$	Anything that R_o something is an A.	OSE	
$\mathbf{Rng}(R_o, A)$	Anything that something R_o is an A.	OSE	
$\mathbf{Dom}(R_d, A)$	Anything that R_d some value is an A.	OSE	
$\mathbf{Rng}(R_d, D_r)$	Any value that something R_d is a D_r.	OSE	
$R_o \sqsubseteq S_o$	If X R_o Y then X S_o Y.	OSE, SOS	
$R_o \equiv S_o$	The properties "R_o" and "S_o" are equivalent.	OSE, Rabbit	✓
$\mathbf{Dis}(R_o, S_o)$	The properties "R_o"and "S_o" are disjoint.	OSE	✓
$\mathbf{Invs}(R_o, S_o)$	"X R_o Y" means the same as "Y S_o X".	OSE	✓
$\mathbf{Fun}(R_o)$	Everything R_o at most one thing.	OSE, ACE	
$\mathbf{InvFun}(R_o)$	If there is something X then at most one thing R_o X.	ACE	
$\mathbf{Ref}(R_o)$	Everything "R_o" itself.	ACE, SOS	
$\mathbf{Irr}(R_o)$	Nothing "R_o" itself.	ACE, SOS	
$\mathbf{Sym}(R_o)$	"X R_o Y" means the same as "Y R_o X".	OSE	
$\mathbf{Asym}(R_o)$	The property "R_o" is asymmetric.	OSE, Rabbit	
$\mathbf{Tra}(R_o)$	If X R_o Y and Y R_o Z then X R_o Z.	SOE, SOS	
$R_d \sqsubseteq S_d$	If X R_d a value then X S_d that value.	OSE, SOS	
$R_d \equiv S_d$	The properties "R_d" and "S_d" are equivalent.	OSE, Rabbit	✓
$\mathbf{Dis}(R_d, S_d)$	The properties "R_d"and "S_d" are disjoint.	OSE	✓
$\mathbf{Fun}(R_d)$	Everything R_d at most one value.	OSE, ACE	
$a \in A$	a is an A.	OSE, ACE, Rabbit, SOS	
$R_o(a, b)$	a R_o b.	OSE, ACE, Rabbit, SOS	
$R_d(a, l \star D_t)$	a R_d a D_t value of l.	OSE	
$\mathbf{Sam}(a, b)$	a and b are the same individual.	OSE, SOS	✓
$\mathbf{Dif}(a, b)$	a and b are different individuals.	OSE, SOS	✓

these theories.

According to Johnson-Laird and Bara [JLB84, JLB91], in syllogistic inferences (i.e., inferences consist of two premises and a conclusion, each of which is in one of the four following forms: "All A are B", "Some A are B", "No A are B", and "Some A are not B", where A and B are two categorical terms such as 'human' and 'mortal') as well as three-term relational inferences (i.e., inferences consist of two premises and a conclusion in which the premises are in the form of "ARB" and "BRC", and the conclusion is of the form "ARC", where A and B are two categorical terms, and R is a binary relation such as "is taller than"), the arrangement of arguments within the premises—called the *figure* of the premises—affects the form of conclusions that subjects draw, their performance accuracy, and the speed of response on *conclusion-production* tasks (i.e., the tasks in which the premises are given, and subjects are asked to formulate the conclusion themselves). Additionally, the figure "AB-BC" is the easiest one for most people to reason with. Given two premises whose figure is "AB-BC", the subjects tend to elicit conclusions that have the figure of "AC". Similarly, given two premises whose figure is "BA-CB", the subjects tend to elicit conclusions that have the figure of "CA". These phenomena of deductive reasoning with syllogisms are called "*figural effects*". According to the authors, the figural effects are due to the fact that human reasoners tend to rearrange the figure to make the two occurrences of the middle term contiguous (e.g., "AB-BC" in the former case, and "CB-BA" in the latter). Because of the "first in first out" principle of working memory, reasoners in the latter case need to recall the first premise while working with the second. Therefore, the latter case is more difficult than the former.

For modus ponens (i.e., "If P then Q; P; therefore Q" where P and Q are two propositional statements) and modus tollens (i.e., "If P then Q; $\neg Q$; therefore $\neg P$" where P and Q are two propositional statements) in which figural effects are not applicable, Girotto et al. [GMT97] conducted a number of empirical studies based on the conclusion-production paradigm which confirmed that the premise order affects the subjects' performance on modus tollens, but not modus ponens tasks. Specifically, subjects' performance accuracy on the inverted modus tollens "$\neg Q$; if P then Q; therefore $\neg P$" is significantly higher than that on the traditional modus tollens. On the other hand, subjects' performance accuracies on the two modus ponens "If P then Q; P; therefore Q" and "P; if P then Q; therefore Q" are equally high. According to Girotto et al., the participants perform better

on the inverted modus tollens because they are forced to consider the crucial case of the failure "$\neg Q$" *before* working with the conditional statement.

As opposed to the conclusion-production paradigm, the *conclusion-evaluation* paradigm is the one in which both the premises and the conclusion of an inference are given, and the participants are asked whether they can understand this inference. Although this paradigm is less popular than the former one, it has also been intensively studied in psychological research areas [MEH04, SB07]. In our work, we are interested in whether human reasoners can understand a given verbalisation for a rule. More precisely, given both the premises and the conclusion of a rule, presented in English text, we are interested in whether the reasoners can understand the relevant inference. This analysis suggests that the conclusion-evaluation paradigm is more relevant for our purposes.

Morley et al. [MEH04] found that the figure of premises affects the subjects' performance accuracy in conclusion-production tasks but *not* in conclusion-evaluation tasks. According to the authors, the presented conclusion in the latter tasks guides the inferential process, but in the former tasks there are no conclusions to guide this process. More recently, Stupple and Ball [SB07] found that although the figure "AB-BC" has no effects on subjects' performance accuracy on conclusion-evaluation tasks, it helps to reduce their inspection times on the premises as well as their processing effort. In order words, presenting the premises of a conclusion-evaluation task in the figure of "AB-BC" can help to speed up subject's reading time as well as reduce their effort.

To identify the most understandable verbalisations for our rules, first the rule set was thoroughly examined. In this examination, the premise order, the figure of the premises were analysed, and the theoretical insights from the psychology of reasoning described above were employed as the key criteria for identifying the most understandable verbalisations. When these criteria provided no guidance, we used an empirical test to find out which verbalisations would be understood best by the participants. This section describes the manual examination of our rule set. The empirical test and its results are presented in the subsequent section.

6.3.1 Selection of Verbalisations

Through our examination, the most understandable verbalisations of 42 (out of 57) deduction rules are identified. These rules are listed as 1-42 in Table 6.3[4]. Single-premise rules are first examined. Among 11 rules of this type, 7 of them contain no commutative constructors, and so have a unique verbalisation. These rules are listed from 1-7 in Table 6.3.

For rules that consist of two or more premises, their figures of premises are examined. Unlike the syllogistic and relational inferences (used in Johnson-Laird and Bara's experiments [JLB84]) in which premises contain only terms as arguments, our rules consist of premises that contain a wide variety of arguments, including class names, object property names, data property names, etc. Thus, the figures for each argument type are computed and analysed separately.

Rules 'SubCls-SubCls-1' and 'SubObj-SubObj' (listed at 8 and 9 in Table 6.3) are the only cases that have only one type of arguments, and the figures of their premises conform to the standard figure "AB-BC". Rules 10-12 in Table 6.3 have more than one type of arguments, but the figures of their class arguments[5]—their main argument type—conform to the standard figure. Hence, their best verbalisations can be easily identified.

Rule 'ObjSom-Bot-1' (rule 13 in Table 6.3), which infers $X \sqsubseteq \bot$ from $X \sqsubseteq \exists R_o.Y$ and $Y \sqsubseteq \bot$ is a special case. The figure of its premises does not conform to the standard figure. However, there is a similarity between this rule and the modus tollens of which the best premise order was identified by Girotto et al. [GMT97]. If the premise $X \sqsubseteq \exists R_o.Y$ ("Every X is a Y") is presented before $Y \sqsubseteq \bot$ ("Nothing is a Y"), the former will cause a presupposition that instances of both classes X and Y already exist—this contradicts the latter premise, and may confuse the readers. To cancel this presupposition, the premise $Y \sqsubseteq \bot$ needs to be presented first in order to emphasize the non-existence of Y's instances from the start. A similar approach is applied to the rules 'ObjSom-Bot-1', 'ObjDom-Bot' and 'ObjRng-Bot' (rules 14-16 in Table 6.3).

For rules 17-26 in Table 6.3, the figures of their main arguments—class arguments in the the first two rules and property arguments in the remaining rules—do not conform to the

[4]The ordering of deduction rules in Table 6.3 are different from that in Table 4.3

[5]In rule 12 in Table 6.3, D_r is a data range but treated as a class argument in the analysis of the figure of its premises.

standard figure. However, these rules are quite similar to the modus ponens discussed in Girotto et al.'s work [GMT97] in the effect of premise order—that is, both ways of ordering their premises have little or no effect on understanding the relevant inferences. Therefore, a common figure of 'A-AB' (or '$R_o - R_oS_o$' for object property arguments) is selected. Rule 27 is quite similar to rules 17-26, but the object property R_o is presented twice in its verbalisation. This axiom is presented first in the best verbalisation of this rule.

On the contrary, we stipulate presenting $\mathbf{Tra}(R_o)$ as the last premise in rules 28 and 29. The two $\sqsubseteq$ axioms in rule 29 are then sorted by their own figure of arguments. In rules 30 and 31 in Table 6.3, we stipulate presenting the axiom containing the constructor $\sqcup$ at the beginning. The other two premises in rule 31 are sorted based on the order of occurrences of the arguments U and V in the first premise.

In rules 32-42 in Table 6.3, the figures of premises of both types (class and property arguments), however, do not conform to any of the above-mentioned figures. In fact, it is impossible to order the arguments in these rules in such a way that yields contiguous occurrences of the middle terms. Example figures of these rules are "AB-AC", "AB-CB", "AB-AB", etc. It can be seen that the premises of these rules are independent, and the order in which they are presented would have no or very little effect on subjects' performance. Therefore, we pickup the best ones based on our own principle—that is, first presenting affirmative or simple information, then negative or more complex information. The best verbalisations for rules 32 to 40 are selected according to this principle. Rules 41 and 42 are extended from rules 39 and 40, respectively, in which the first premises are replaced by a number of relevant premises, and these premises are sorted in a relevant way.

6.3.2 Extra Statements for Implicit Information

Among the collected deduction rules, there exist rules in which the conclusion is inferred from not only the premises but also *implicit* information in OWL. Specifically, in rules 41, 46, 47, and 48 in Table 6.3, the unsatisfiability in the conclusion is caused by the disjointness between two data types presented in the premises (e.g., the disjointness between String and Integer data types), but this information is implicit in OWL and not stated

Table 6.3: Deduction rules and their candidate verbalisations for the premises and entailment; verbalisations selected through the empirical study are marked with *. In this table, 'Figure' means the figure of arguments in the associated verbalisation, 'C' means the figure of class arguments, 'P' means the figure of property arguments, and '__' means no relevant arguments.

ID	Name	Deduction Rule	Verbalisations	Figure
1	ObjInt-2	$X \sqsubseteq Y \sqcap Z[\sqcap \ldots]$ $\rightarrow X \sqsubseteq Y$	Every X is both a Y and a Z. →Every X is a Y.	None
2	ObjUni-2	$Y \sqcup Z[\sqcup \ldots] \sqsubseteq X$ $\rightarrow Y \sqsubseteq X$	Everything that is a Y or a Z is an X. →Every Y is an X.	None
3	ObjExt	$X \sqsubseteq= n_1 R_o.Y$ $\rightarrow X \sqsubseteq\geq n_2 R_o.Y$ $(n_1 \geq n_2 > 0)$	Every X R_o exactly n_1 Y(s). →Every X R_o at least n_2 Y(s).	None
4	Top	$\top \sqsubseteq X$ $\rightarrow Y \sqsubseteq X$	Everything is an X. →Every Y is an X.	None
5	Bot	$X \sqsubseteq \bot$ $\rightarrow X \sqsubseteq Y$	Nothing is an X. →Every X is a Y.	None
6	ObjCom-1	$X \sqsubseteq \neg X$ $\rightarrow X \sqsubseteq \bot$	Every X is something that is not an X. →Nothing is an X.	None
7	ObjCom-2	$\neg X \sqsubseteq Y$ $\rightarrow \top \sqsubseteq X \sqcup Y$	Everything that is not an X is a Y. →Everything is an X or a Y.	None
8	SubCls-SubCls-1	$X \sqsubseteq Y$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	Every X is a Y. Every Y is a Z. →Every X is a Z.	C:$XY - YZ$ $\rightarrow XZ$
9	SubObj-SubObj	$R_o \sqsubseteq S_o$ $\wedge\ S_o \sqsubseteq T_o$ $\rightarrow R_o \sqsubseteq T_o$	If X R_o Y then X S_o Y. If X S_o Y then X T_o Y. →If X R_o Y then X T_o Y.	P:$R_oS_o - S_oT_o$ $\rightarrow R_oT_o$
10	ObjSom-SubCls	$X \sqsubseteq \exists R_o.Y$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq \exists R_o.Z$	Every X R_o a Y. Every Y is a Z. →Every X R_o a Z.	C:$XY - YZ$ $\rightarrow XZ$
11	ObjSom-ObjMin	$X \sqsubseteq \exists R_o.Y$ $\wedge\ \geq 1 R_o.Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	Every X R_o a Y. Everything that R_o at least one Y is a Z. →Every X is a Z.	C:$XY - YZ$ $\rightarrow XZ$
12	DatSom-DatMin	$X \sqsubseteq \exists R_d.D_r$ $\wedge\ \geq 1 R_d.D_r \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	Every X R_d a D_r. Everything that R_d at least one D_r is a Z. →Every X is a Z.	C:$XD_r - D_rZ$ $\rightarrow XZ$
13	ObjSom-Bot-1	$X \sqsubseteq \exists R_o.Y$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$	Nothing is a Y. Every X R_o a Y. →Nothing is an X.	C:$Y - XY$ $\rightarrow X$
14	ObjSom-Bot-2	$X \sqsubseteq \exists R_o.Y$ $\wedge\ Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$	Nothing is a Y. Every X R_o a Y. →Nothing is an X.	C:$Y - XY$ $\rightarrow X$
15	ObjDom-Bot	$\mathbf{Dom}(R_o, X)$ $\wedge\ X \sqsubseteq \bot$ $\rightarrow \top \sqsubseteq \forall R_o.\bot$	Nothing is a X. Anything that R_o something is an X. →Everything that R_o nothing at all is an X.	C:$X - X$ $\rightarrow X$
16	ObjRng-Bot	$\mathbf{Rng}(R_o, X)$ $\wedge\ X \sqsubseteq \bot$ $\rightarrow \top \sqsubseteq \forall R_o.\bot$	Nothing is a X. Anything that something R_o is an X. →Everything that R_o nothing at all is an X.	C:$X - X$ $\rightarrow X$
17	ObjDom-SubCls	$\mathbf{Dom}(R_o, X)$ $\wedge\ X \sqsubseteq Y$ $\rightarrow \mathbf{Dom}(R_o, Y)$	Anything that R_o something is an X. Every X is a Y. →Anything that R_o something is a Y.	C:$X - XY$ $\rightarrow Y$
18	ObjRng-SubCls	$\mathbf{Rng}(R_o, X)$ $\wedge\ X \sqsubseteq Y$ $\rightarrow \mathbf{Rng}(R_o, Y)$	Anything that something R_o is an X. Every X is a Y. →Anything that something R_o is a Y.	C:$X - XY$ $\rightarrow Y$
19	ObjDom-SubObj	$\mathbf{Dom}(R_o, X)$ $\wedge\ S_o \sqsubseteq R_o$ $\rightarrow \mathbf{Dom}(S_o, X)$	Anything that R_o something is an X. If X S_o Y then X R_o Y. →Anything that S_o something is an X.	P:$R_o - S_oR_o$ $\rightarrow S_o$

Continued on Next Page...

20	ObjRng-SubObj	$\mathbf{Rng}(R_o, X)$ $\wedge\ S_o \sqsubseteq R_o$ $\rightarrow \mathbf{Rng}(S_o, X)$	Anything that something R_o is an X. If X S_o Y then X R_o Y. →Anything that something S_o is an X.	P:$R_o - S_o R_o$ $\rightarrow S_o$
21	ObjRng-ObjInv	$\mathbf{Rng}(R_o, X)$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Dom}(S_o, X)$	Anything that something R_o is an X. "X R_o Y" means the same as "Y S_o X". →Anything that S_o something is an X.	P:$R_o - R_o S_o$ $\rightarrow S_o$
22	ObjDom-ObjInv	$\mathbf{Dom}(R_o, X)$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Rng}(S_o, X)$	Anything that R_o something is an X. "X R_o Y" means the same as "Y S_o X". →Anything that something R_o is an X.	P:$R_o - R_o S_o$ $\rightarrow S_o$
23	ObjRng-ObjSym	$\mathbf{Rng}(R_o, X)$ $\wedge\ \mathbf{Sym}(R_o)$ $\rightarrow \mathbf{Dom}(R_o, X)$	Anything that something R_o is an X. "X R_o Y" means the same as "Y R_o X". →Anything that is R_o something is an X.	P:$R_o - R_o R_o$ $\rightarrow R_o$
24	ObjDom-ObjSym	$\mathbf{Dom}(R_o, X)$ $\wedge\ \mathbf{Sym}(R_o)$ $\rightarrow \mathbf{Rng}(R_o, X)$	Anything that R_o something is an X. "X R_o Y" means the same as "Y R_o X". →Anything that something R_o is an X.	P:$R_o - R_o R_o$ $\rightarrow R_o$
25	ObjSom-SubObj	$X \sqsubseteq \exists R_o.Y$ $\wedge\ R_o \sqsubseteq S_o$ $\rightarrow X \sqsubseteq \exists S_o.Y$	Every X R_o a Y. If X R_o Y then X S_o Y. →Every X S_o a Y.	P:$R_o - R_o S_o$ $\rightarrow S_o$
26	ObjAll-ObjInv	$X \sqsubseteq \forall R_o.Y$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \exists S_o.X \sqsubseteq Y$	Every X R_o only Ys. "X R_o Y" means the same as "Y S_o X". →Everything that S_o an X is a Y.	P:$R_o - R_o S_o$ $\rightarrow S_o$
27	ObjTra-ObjInv	$\mathbf{Tra}(R_o)$ $\wedge\ \mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Tra}(S_o)$	If X R_o Y and Y R_o Z then X R_o Z. "X R_o Y" means the same as "Y S_o X". →If X S_o Y and Y S_o Z then X S_o Z.	P:$R_o R_o - R_o S_o$ $\rightarrow S_o$
28	ObjSom-ObjTra	$X \sqsubseteq \exists R_o.(\exists R_o Y)$ $\wedge\ \mathbf{Tra}(R_o)$ $\rightarrow X \sqsubseteq \exists R_o.Y$	Every X R_o something that R_o a Y. If X R_o Y and Y R_o Z then X R_o Z. →Every X R_o a Y.	C:$R_o R_o - R_o$ $\rightarrow R_o$
29	ObjSom-ObjSom-ObjTra	$X \sqsubseteq \exists R_o.Y$ $\wedge\ Y \sqsubseteq \exists R_o.Z$ $\wedge\ \mathbf{Tra}(R_o)$ $\rightarrow X \sqsubseteq \exists R_o.Z$	Every X R_o a Y. Every Y R_o a Z. If X R_o Y and Y R_o Z then X R_o Z. →Every X R_o a Z.	C:$XY - YZ - \_\_$ $\rightarrow XZ$
30	ObjUni-SubCls	$X \sqsubseteq (Y \sqcup Z)$ $\wedge\ Y \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	Every X is a Y or a Z. Every Y is a Z. →Every X is a Z.	C:$XYZ - YZ$ $\rightarrow XZ$
31	ObjUni-SubCls-SubCls	$X \sqsubseteq (U \sqcup V)$ $\wedge\ U \sqsubseteq Z$ $\wedge\ V \sqsubseteq Z$ $\rightarrow X \sqsubseteq Z$	Every X is a U or a V. Every U is a Z. Every V is a Z. →Every X is a Z.	C:$XYZ - YW - ZW$ $\rightarrow XW$
32	SubCls-SubCls-2	$X \sqsubseteq Y$ $\wedge\ X \sqsubseteq Z$ $\rightarrow X \sqsubseteq Y \sqcap Z$	Every X is a Y. Every X is a Z. →Every X is both a Y and a Z.	C:$XY - XZ$ $\rightarrow XYZ$
33	ObjMin-ObjMax	$X \sqsubseteq \geq n_1 R_o.Y$ $\wedge\ X \sqsubseteq \leq n_2 R_o.Y,\ n_2 < n_1$ $\rightarrow X \sqsubseteq \bot$	Every X R_o at least n_1 Y(s). Every X R_o at most n_2 Y(s). →Nothing is an X.	C:$XY - XY$ $\rightarrow X\bot$
34	SubCls-ObjCom-1	$X \sqsubseteq Y$ $\wedge\ X \sqsubseteq \neg Y$ $\rightarrow X \sqsubseteq \bot$	Every X is a Y. Every X is not a Y. →Nothing is an X.	C:$XY - XY$ $\rightarrow X\bot$
35	SubCls-ObjCom-2	$X \sqsubseteq Y$ $\wedge\ \neg X \sqsubseteq Y$ $\rightarrow \top \sqsubseteq Y$	Every X is a Y. Everything that is not an X is a Y. →Everything is a Y.	C:$XY - XY$ $\rightarrow \top X$
36	ObjDom-ObjAll	$\mathbf{Dom}(R_o, X)$ $\wedge\ \forall R_o.\bot \sqsubseteq X$ $\rightarrow \top \sqsubseteq X$	Anything that R_o something is an X. Everything that R_o nothing at all is an X. →Everything is an X.	:$\top X - \bot X$ $\rightarrow \top X$
37	ObjSom-ObjAll-1	$\exists R_o.Y \sqsubseteq X$ $\wedge\ \forall R_o.\bot \sqsubseteq X$ $\rightarrow \forall R_o.Y \sqsubseteq X$	Everything that R_o a Y is an X. Everything that R_o nothing at all is an X. →Everything that R_o only Ys is an X.	C:$XY - \bot Y$ $\rightarrow XY$
38	ObjSom-ObjAll-2	$X \sqsubseteq \exists R_o.\top$	Every X R_o something.	C:$X\top - XY$

Continued on Next Page...

		$\wedge\ X \sqsubseteq \forall R_o.Y$ $\rightarrow X \sqsubseteq \exists R_o.Y$	Every X R_o only Ys. →Every X R_o a Y.	$\rightarrow XY$
39	DatMin-DatFun	$X \sqsubseteq \geq nR_d.D_r$, $n > 1$ $\wedge$ **Fun**($\mathbf{R_d}$) $\rightarrow X \sqsubseteq \bot$	Every X R_d at least n D_rs. Everything R_d at most one value. →Nothing is an X.	C:__ − XD_r $\rightarrow X\bot$ P:$R_d - R_d$ $\rightarrow X\bot$
40	ObjMin-ObjFun	$X \sqsubseteq \geq nR_o.Y$, $n > 1$ $\wedge$ **Fun**($\mathbf{R_o}$) $\rightarrow X \sqsubseteq \bot$	Every X R_o at least n Y(s). Everything R_o at most one thing. →Nothing is an X.	C:__ − XY $\rightarrow X\bot$ P:$R_o - R_o$ $\rightarrow$ __
41	DatVal-DatVal-DatFun	$X \sqsubseteq \exists R_d.\{l_0 \star D_{t0}\}$ $\wedge\ X \sqsubseteq \exists R_d.\{l_1 \star D_{t1}\}$ $\wedge$ **Fun**($\mathbf{R_d}$) D_{t0} & D_{t1} are disjoint or $l_0 \neq l_1$ $\rightarrow X \sqsubseteq \bot$	Every X R_d a D_{t0} value of l_0. Every X R_d a D_{t1} value of l_1. D_{t0} values are not D_{t1} values. Everything R_d at most one value. →Nothing is an X.	C:$X - X$ − __ $\rightarrow X\bot$ P:$R_d - R_d - R_d$ $\rightarrow$ __
42	ObjVal-ObjVal-DifInd-ObjFun	$X \sqsubseteq \exists R_o.\{i\}$ $\wedge\ X \sqsubseteq \exists R_o.\{j\}$ $\wedge$ **Dif**(**i**, **j**) $\wedge$ **Fun**($\mathbf{R_o}$) $\rightarrow X \sqsubseteq \bot$	Every X R_o i. Every X R_o j. i and j are different individuals. Everything R_o at most one thing. →Nothing is an X.	C:$X - X$ − __ − __ $\rightarrow X\bot$ P:$R_d - R_d$ − __ − R_d →__
43	ObjAll	$X \equiv \forall R_o.Y$ $\rightarrow \forall R_o.\bot \sqsubseteq X$	*Verbalisation 1:* An X is anything that R_o only Ys. →Everything that R_o nothing at all is an X. *Verbalisation 2:* Every X R_o only Ys; everything that R_o only Ys is an X. →Everything that R_o nothing at all is an X.	None
44	DisCls-SubCls-SubCls	**Dis**($\mathbf{X}, \mathbf{Y}$) $\wedge\ U \sqsubseteq X$ $\wedge\ V \sqsubseteq Y$ $\rightarrow$ **Dis**(U, V)	*Verbalisation 1:* No X is a Y. Every U is an X. Every V is a Y. →No U is a V. *Verbalisation 2:* Nothing is both an X and a Y. Every U is a X. Every V is a Y. →Nothing is both a U and a V.	None
45	ObjSom-ObjDom	$X \sqsubseteq \exists R_o.Z$ $\wedge$ **Dom**(R_o, Y) $\rightarrow X \sqsubseteq Y$	*Verbalisation 1:* Every X R_o a Z. Anything that R_o something is a Y. →Every X is a Y. *Verbalisation 2:* Anything that R_o something is a Y. Every X R_o a Z. →Every X is a Y.	None
46	DatSom-DatRng	$X \sqsubseteq \exists R_d.D_{r0}$ $\wedge$ **Rng**(R_d, D_{r1}) D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	*Verbalisation 1:* Every X R_d a D_{r0}. Any value that something R_d is a D_{r1}. D_{r0} values are not D_{r1} values. →Nothing is an X. *Verbalisation 2:* Any value that something R_d is a D_{r1}. Every X R_d a D_{r0}. D_{r0} values are not D_{r1} values. →Nothing is an X.	None
47	DatMin-DatRng	$X \sqsubseteq \geq nR_d.D_{r0}$, $n > 0$ $\wedge$ **Rng**(R_d, D_{r1}) D_{r0} & D_{r1} are disjoint	*Verbalisation 1:* Every X R_d at least n D_{r0}(s). Any value that something R_d is a D_{r1}.	None

Continued on Next Page...

		$\rightarrow X \sqsubseteq \bot$	D_{r0} values are not D_{r1} values. →Nothing is an X. *Verbalisation 2:* Any value that something R_d is a D_{r1}. Every X R_d at least n D_{r0}(s). D_{r0} values are not D_{r1} values. →Nothing is an X.	
48	DatVal-DatRng	$X \sqsubseteq \exists R_d.\{l \star D_t\})$ $\wedge$ **Rng**(R_d, D_r) D_t & D_r are disjoint $\rightarrow X \sqsubseteq \bot$	*Verbalisation 1:* Every X R_d a D_t value of l. Any value that something R_d is a D_r. D_t values are not D_r values. →Nothing is an X. *Verbalisation 2:* Any value that something R_d is a D_r. Every X R_d a D_t value of l. D_t values are not D_r values. →Nothing is an X.	None
49	EquCls	$X \equiv Y$ $\rightarrow X \sqsubseteq Y$	*Verbalisation 1:* An X is anything that is a Y. →Every X is a Y. *Verbalisation 2: (*)* Every X is a Y; every Y is an X. →Every X is a Y.	None
50	ObjInt-1	$X \equiv Y \sqcap Z$ $\rightarrow X \sqsubseteq Y$	*Verbalisation 1:* An X is anything that is both a Y and a Z. →Every X is a Y. *Verbalisation 2: (*)* Every X is both a Y and a Z; everything that is both a Y and a Z is an X. →Every X is a Y.	None
51	ObjUni-1	$X \equiv Y \sqcup Z$ $\rightarrow X \sqsubseteq Y$	*Verbalisation 1:* An X is anything that is a Y or a Z. →Every Y is a X. *Verbalisation 2: (*)* Every X is a Y or a Z; everything that is a Y or a Z is an X. →Every Y is a X.	None
52	SubCls-DisCls	$X \sqsubseteq Y$ $\wedge$ **Dis**(X, Y) $\rightarrow X \sqsubseteq \bot$	*Verbalisation 1:(*)* Every X is a Y. No X is a Y. →Nothing is an X. *Verbalisation 2:* Every X is a Y. No Y is an X. →Nothing is an X. *Verbalisation 3:* Every X is a Y. Nothing is both an X and a Y. →Nothing is an X.	None
53	Top-DisCls	$\top \sqsubseteq Y$ $\wedge$ **Dis**(X, Y) $\rightarrow X \sqsubseteq \bot$	*Verbalisation 1:(*)* Everything is a Y. No X is a Y. →Nothing is an X. *Verbalisation 2:* Everything is a Y. No Y is an X. →Nothing is an X.	None

Continued on Next Page...

			Verbalisation 3: Everything is a Y. Nothing is both an X and a Y. →Nothing is an X.	
54	SubCls-SubCls-DisCls	$X \sqsubseteq Y$ $\wedge\ X \sqsubseteq Z$ $\wedge\ \mathbf{Dis}(\mathbf{Y}, \mathbf{Z})$ $\rightarrow X \sqsubseteq \bot$	*Verbalisation 1: (*)* Every X is a Y. Every X is a Z. No Y is a Z. →Nothing is an X. *Verbalisation 2:* Every X is a Y. Every X is a Z. Nothing is both a Y and a Z. →Nothing is an X.	None
55	ObjInt-DisCls	$X \sqsubseteq \exists R_o.(Y \sqcap Z)$ $\wedge\ \mathbf{Dis}(Y, Z)$ $\rightarrow X \sqsubseteq \bot$	*Verbalisation 1: (*)* Every X R_o something that is both a Y and a Z. No Y is a Z. →Nothing is an X. *Verbalisation 2:* Every X R_o something that is both a Y and a Z. Nothing is both a Y and a Z. →Nothing is an X.	None
56	DatSom-DatDom	$X \sqsubseteq \exists R_d.D_r$ $\wedge\ \mathbf{Dom}(R_d, Y)$ $\rightarrow X \sqsubseteq Y$	*Verbalisation 1: (*)* Every X R_d a D_r. Anything that R_d some value is a Y. →Every X is a Y. *Verbalisation 2:* Anything that R_d some value is a Y. Every X R_d a D_r. →Every X is a Y.	None
57	ObjSom-ObjRng	$X \sqsubseteq \exists R_o.Y$ $\wedge\ \mathbf{Rng}(R_o, Z)$ $\rightarrow X \sqsubseteq \exists R_o.(Y \sqcap Z)$	*Verbalisation 1: (*)* Every X R_o a Y. Anything that something R_o is a Z. →Every X R_o something that is both a Y and a Z. *Verbalisation 2:* Anything that something R_o is a Z. Every X R_o a Y. →Every X R_o something that is both a Z and a Y.	None

explicitly in the premises. This means that unless the readers are aware of this information in advance, they cannot understand the unsatisfiability in the conclusion.

To help ordinary users understand those rules more easily, an extra sentence that explicitly describes the disjointness between the two data types is inserted into the verbalisation of each rule. This sentence is always inserted after the last premise in the verbalisation. Given two disjoint data types 'integer' and 'decimal', for instance, where the data type 'integer' is mentioned somewhere before the data type 'decimal' in the rule, the extra sentence to describe this disjointness is "No integer values are decimal values". If the data type 'decimal' is mentioned before the data type 'integer' then the sentence "No decimal values are integer values" would be inserted.

In fact, even when the two data types in rule 41 in Table 6.3 are not disjoint, there exists another cause of unsatisfiability in this rule—that is, the difference between two literal values presented in the rule—and this information is also left implicit to the readers. However, this information is quite trivial (e.g., 3.2 and 5.1 are different values); thus no extra statements would be added into the verbalisation of the rule.

6.4 An Empirical Study

As discussed in Section 6.3.1, we could identify the most understandable verbalisations of 42 (out of 57) rules by relying on existing theoretical insights from the psychology of reasoning, and only those of the remaining 15 rules, listed as 43-57 in Table 6.3, needed to be identified empirically. However, on closer inspection we found that only 9 of them would need to be tested, and testing the remaining rules would bring no or little advantage[6]. Horridge et al. [HPS09b, Hor11] proved that the inference in rule 43 is very difficult even for OWL experts, and the major source of its difficulty does not lie in the verbalisation of the constructor $\equiv$, but in the trivial satisfiability of the universal restriction in OWL. Therefore, we excluded this rule from our study.

In rule 44, the main source of multiple verbalisations lies in the commutativity of the constructor **Dis**. However, since this constructor appears in both a premise and the conclusion, and these axioms should be verbalised similarly, only two candidate verbalisations

[6]Testing all of these 15 deduction rules empirically would be a better solution. However, due to the practical limitation of this PhD project, we only tested those that had higher priorities

are considered here, as shown in Table 6.3. Since these verbalisations are symmetric, testing them would bring no or little help in verifying how they can affect subjects' performance on the rule. Therefore, this rule was also excluded in our study.

Rule 45 is similar to rule 56. They are about the use of the constructor **Dom**, but the former is for object properties while the later is for data properties. Rules 46-48 are similar to each other, and all of them are very close to rule 57 because they are about the use of the constructor **Rng** for both object and data properties. Additionally, their main source of the multiplicity of verbalisations lies in the ordering of the premises—that is, it is unclear which order of the premises would facilitate human understanding. Therefore, two representatives of these rules, namely rules 56 and 57, were randomly selected for this study. In total, we tested the 9 rules from 49-57 in Table 6.3.

6.4.1 Candidate Verbalisations

This sub-section describes the identification of candidate verbalisations for each tested rule. As discussed in Section 6.3, the number of all possible verbalisations of a rule is often non-trivial, but we can rule out some inferior verbalisations by analysing the figure of their premises, in the same way as we did in the manual examination. Particularly, in rules 49-51, there is only one premise, and their major source of the multiplicity of verbalisations lies in the multiplicity of verbalisations of the constructor $\equiv$; thus, they all have exactly two candidate verbalisations, as shown in Table 6.3.

Rules 52-55 have multiple premises, but their optimal orderings can be identified based on existing theoretical insights from psychology as well as our own principle of presenting affirmative and simple information first, as shown in Table 6.3. The major source of the multiplicity of their verbalisations is the commutativity property and the verbalisation multiplicity property of the constructor **Dis**. Therefore, each of these rules have 2*2 or 4 candidate verbalisations. However, in rules 52-55, the two verbalisations for the axiom $\mathbf{Dis}(X, Y)$ "Nothing is both an X and a Y" and "Nothing is both a Y and an X" make little or no difference in the rules' verbalisations; therefore, only three candidate verbalisations were selected. The two verbalisations "No X is a Y" and "No Y is an X" in rules 54 and 55 also make little or no difference in the rules' verbalisations; therefore, only two of them were selected as candidate verbalisations. As an example, the selection of candidate

verbalisations for rule 53 which originally has 8 possibilities (as shown in Table 6.1) can be explained as follows:

> Firstly, the verbalisations 5-8 can be eliminated because their premise orderings are worse than those of the verbalisations 1-4, according to our principle of presenting affirmative and simple information first. Among the four remaining verbalisations, 2 and 4 only differ in the arrangement of arguments in the constructor **Dis**. However, this difference brings no advantage in the verbalisation of the rule; thus, we can further eliminate one of them, for instance, the verbalisation 4. As a result, only 3 of 8 possibilities were selected as candidate verbalisations for this rule in our study.

Finally, rules 56 and 57 have two premises, and the premise ordering is the major source of their multiplicity of verbalisations. Two candidate verbalisations were created for each of these rules.

6.4.2 Materials

We devised a *deduction problem* for each candidate verbalisation. In each deduction problem, the premises of the rule were given, the subjects were asked to answer whether each of *four* given statements followed from the premises. Among these statements, the entailment of the rule was always included, and there might be more than one statements following from the premises. We awarded a mark of 0.25 for each correctly classified statement (as either "Follows" or "Does not Follow"), so 1.0 if all four statements were classified correctly. Both the premises and the four statements were given in English, replacing individual, class, and property variables by fictional names, nouns, and verbs[7] so that the readers would not be biassed by domain knowledge. We called these problems *test problems*. There were 20 test problems corresponding to 20 candidate verbalisations, and so a total of 20*4 or 80 *test questions*. For balancing, 41 "Follows" and 39 "Does not Follow" questions were created.

[7]Fictional words are nonsense words selected from various sources, such as Lewis Carroll's Jabberwocky poem (`http://en.wikipedia.org/wiki/Jabberwocky`), creatures' names in Dungeons and Dragons (`http://en.wikipedia.org/wiki/Category:Dungeons_%26_Dragons_creatures`), an automatic generator (`http://www.soybomb.com/tricks/words/`), and so on.

6.4.3 Method

The study was conducted on CrowdFlower, a crowdsourcing service that allowed customers to upload tasks to be passed to labour channel partners such as Amazon Mechanical Turk[8]. We set up the operation so that the tasks were channelled only to Amazon Mechanical Turk, and were restricted to subjects from Australia, the United Kingdom, and the United States since we were aiming to recruit as many (self-reported) native speakers of English as possible.

To eliminate responses from 'scammers' (people who responded casually without considering the problem seriously), we used CrowdFlower's quality control service which was based on *gold-standard data*: we provided problems called *gold units* for which the correct answers were specified, allowing CrowdFlower to filter automatically any subjects whose performance on gold units fell below a threshold (75%). The management of these gold units was internal to CrowdFlower, and the order for which these gold units would be presented varies randomly on subjects. A weakness of this design was that the total number of problems would be increased.

To limit the total number of problems in our study, we adapted the use of gold units but in a slightly different way. Instead of creating gold units that resembled the test problems, we created gold units as part of the test problems. Specifically, we introduced *two* additional questions into each test problem called *control questions*. Our control questions were designed to resemble test questions but were obvious to any subjects who did the test seriously (rather than responding casually without reading the problem properly). We created two types of control questions called *non-entailment* and *trivial* questions.

A non-entailment question was one in which the statement was about an object that was not mentioned in the premises. The correct answer for non-entailment questions was "Does not follow", trivially. A trivial question was one in which the statement was actually a repetition of one of the premises, so the correct answer was, also trivially, "Follows". Answers of control questions were not counted in the analysis, but used by CrowdFlower to filter spam responses—i.e., for a test problem selected as a gold unit, a response would be accepted only if the answers for both of the control questions were correct. We created a non-entailment and a trivial control question for each test problem (i.e., 20 non-entailment

[8]See `http://crowdflower.com/` and `http://www.mturk.com/` for details.

questions and 20 trivial questions in total), so the correct answers of the control questions were always "Does not Follow" and "Follows", with question order varying so that the non-entailment question sometimes preceded the trivial, and sometimes followed it. For our purpose, 4 of 20 test problems were selected as gold units.

It is important to note that in CrowdFlower subjects are not required to complete all problems. They can give up whenever they want, and their responses will be accepted so long as they perform well on gold units. CrowdFlower randomly assigns test problems to subjects until it collects up to a specified number of valid responses for each problem. In our study we specifed 50. However, since we were only interested in responses in which all 20 test problems were answered, we selected only 46 valid responses.

6.4.4 Results

The main aim of the study was to collect frequency data on whether people understood one verbalisation for a deduction rule better than the others. However, these data would provide a valid estimate of the understandability of verbalisations only if we could control for positive response bias: in the extreme case, a subject that always gave the positive answer ("Follows" rather than "Does not follow") would get all test questions right, regardless of their difficulty[9]. We used control questions to address this issue—additional to the CrowdFlower filtering based on 4 gold units—i.e., we analysed the responses of control questions in all test problems to check whether a subject was taking the survey seriously.

In the main analysis, the responses of test questions were analysed in two ways. We analysed the responses of all test questions for a deduction rule in order to identify *its best verbalisation*—i.e., the one that was understood best by the subjects. We also analysed the responses of all test questions for a group of rules in order to identify *the best wording for an OWL axiom*. Specifically, we analysed the responses for rules 49-51 in Table 6.3 to identify the best wording for the axiom constructor $\equiv$, rules 52-55 for the axiom constructor **Dis**, and rules 56 and 57 for the best premise ordering for the axiom constructors **Dom** and **Rng**.

Control Questions

[9]Negative response bias would not need to be controlled: a subject gave the negative answer ("Does not Follow" rather than "Follows") because s/he did not understand the inference correctly, or found it too difficult to understand.

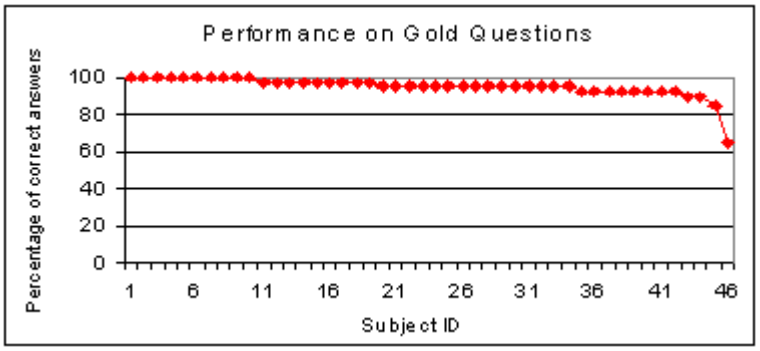


Figure 6.1: Subjects' performance on the control problems, sorted decreasingly

Figure 6.1 shows that for the 46 subjects that participated in our study, there was one subject who answered only 65% of the control questions correctly, suggesting that s/he was not performing the test seriously, so the corresponding response was accordingly discarded. Of the 45 remaining subjects, the proportions of correct answers on control questions were 85% and higher, suggesting that they were performing the task seriously.

Response Bias

Table 6.4 shows the absolute frequencies of the responses "Follows" (+F) and "Does not follow" (−F) for all questions in the study—control as well as test. It also subdivides these frequencies according to whether the answer was correct (+C) or incorrect (−C). Thus for example the cell +F+C counts cases in which subjects answered "Follows" when this was the correct answer, while +F−C counts cases in which they answered "Follows" when this was incorrect.

Recalling that our study consisted of 20*4 or 80 test questions and 20*2 or 40 control questions. Among these questions, the correct answer of 41 test questions and 20 control questions was "Follows", so the percentage of +F answers for a subject that always answered correctly would be (41+20)/(80+40)=51%. If subjects had a positive response bias, we would expect an overall rate higher than this, but in fact we obtained 2504/5400 or 46%, suggesting little or no positive response bias.

Looking at the distribution of incorrect answers, we can also ask whether subjects erred through being too ready to accept invalid conclusions (+F−C), or too willing to reject conclusions that were in reality valid (−F−C). The table shows a clear tendency towards the latter, with 671 responses in −F−C comparing with an expected value of 2896*1101/5400 or 590 calculated from the overall frequencies. In other words, subjects were more likely

Table 6.4: Distribution of subject's responses—i.e., "Follows" (+F) and "Does not Follow" (−F)—according to their correctness—i.e., "Correct" (+C) and "Incorrect" (−C)—in the empirical study

	+F	**−F**	**TOTAL**
+C	2074	2225	4299
−C	430	671	1101
TOTAL	2504	2896	5400

Table 6.5: Distribution of correct answers on two verbalisations for the deduction rule 'EquCls' (rule 49 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** An X is anything that is a Y.	**Verbalisation 2** Every X is a Y; every Y is an X.	**TOTAL**
All Xs are Ys.(*)	39	44	83
All Ys are Xs.(*)	29	44	73
Some but not all Xs are Ys.	40	45	85
Some but not all Ys are Xs.	32	44	76
TOTAL	140	177	317

to err by rejecting a valid conclusion than by accepting an invalid one, a finding confirmed statistically by the significant association between response (±F) and correctness (±C) on a 2×2 chi-square test (χ^2=29.8, df=1, p<0.0001).

The Best Verbalisations for Deduction Rules

Table 6.5 shows the absolute frequencies of correct answers on two candidate verbalisations of rule 49 in Table 6.3. It also subdivides these frequencies by statements used to test the verbalisations. A Wilcoxon Signed Ranks test revealed a statistically reliable difference in subjects' performance between the two verbalisations (Z=3.93, p<0.0001), and the second wording was easier than the first one. This wording is currently used by logicians and in ACE [KF07].

Table 6.6 shows the absolute frequencies of correct answers on two candidate verbalisations of rule 50 in Table 6.3. A Wilcoxon Signed Ranks test revealed no reliable difference in subjects' performance between the two verbalisations (Z=1.34, p=0.18). Similarly, Wilcoxon Signed Ranks tests showed that there were no reliable differences in subjects' performance between two verbalisations in rules 51 (Z=1.36, p=0.17), 54 (Z=1.11, p=0.26), 55 (Z=0.53, p=0.59), 56 (Z=0.56, p=0.57), and 57 (Z=1.10, p=0.27). The absolute frequencies of correct answers of these deductions are shown in Tables 6.7 (rule 51), 6.8 (rule 54),

Table 6.6: Distribution of correct answers on two verbalisations for the deduction rule 'ObjInt-1' (rule 50 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** An X is anything that is both a Y and a Z.	**Verbalisation 2** Every X is both a Y and a Z; everything that is both a Y and a Z is an X.	**TOTAL**
Every X is a Y.(*)	32	41	73
Every Y is an X.	39	35	74
Every X is a Z.(*)	29	41	70
Every Z is an X.	39	34	73
TOTAL	139	151	290

Table 6.7: Distribution of correct answers on two verbalisations for the deduction rule 'ObjUni-1' (rule 51 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** An X is anything that is a Y or a Z.	**Verbalisation 2** Every X is a Y or a Z; everything that is a Y or a Z is an X.	**TOTAL**
Every Y is an X.(*)	33	43	76
Every X is a Y.	39	31	70
Every Z is an X.(*)	33	43	76
Every X is a Z.	35	35	70
TOTAL	140	152	292

Table 6.8: Distribution of correct answers on two verbalisations for the deduction rule 'SubCls-SubCls-DisCls' (rule 54 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** Every X is a Y. Every X is a Z. No Y is a Z.	**Verbalisation 2** Every X is a Y. Every X is a Z. Nothing is both a Y and a Z.	**TOTAL**
Nothing is a X.(*)	21	29	50
Every X is both a Y and a Z.(*)	35	31	66
Every Y is something that is not a Z.(*)	34	25	59
Everything that is not a Z is a Y.	38	37	75
TOTAL	128	122	250

Table 6.9: Distribution of correct answers on two verbalisations for the deduction rule 'ObjInt-DisCls' (rule 55 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** Every X r0 something that is both a Y and a Z. No Y is a Z.	**Verbalisation 2** Every X r0 something that is both a Y and a Z. Nothing is both a Y and a Z.	**TOTAL**
Nothing is an X.(*)	12	17	29
Every X r0 nothing at all.	21	18	39
Every Y is something that is not a Z.(*)	33	29	62
Everything that is not a Z is a Y.	42	42	84
TOTAL	108	106	214

Table 6.10: Distribution of correct answers on two verbalisations for the deduction rule 'DatSom-DatDom' (rule 56 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** Every X R_d a D_r. Anything that R_d some value is a Y.	**Verbalisation 2** Anything that R_d some value is a Y. Every X R_d a D_r.	**TOTAL**
Every X is a Y.(*)	40	40	80
Every Y is an X.	35	34	69
Some but not all Xs are Ys.	41	40	81
Some but not all Ys are Xs.	14	12	26
TOTAL	130	126	256

Table 6.11: Distribution of correct answers on two verbalisations for the deduction rule 'ObjSom-ObjRng' (rule 57 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** Every X R_o a Y. Anything that something R_o is a Z.	**Verbalisation 2** Anything that something R_o is a Z. Every X R_o a Y.	**TOTAL**
Every X R_o something that is both a Y and a Z.(*)	28	27	55
Every X R_o nothing at all.	45	44	89
Every Y is a Z.	16	13	29
Every Z is a Y.	38	37	75
TOTAL	127	121	248

Table 6.12: Distribution of correct answers on three verbalisations for the deduction rule 'SubCls-DisCls' (rule 52 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** Every X is a Y. No X is a Y.	**Verbalisation 2** Every X is a Y. No Y is a X.	**Verbalisation 3** Every X is a Y. Nothing is both an X and a Y.	**TOTAL**
Nothing is an X.(*)	26	22	30	78
Nothing is a Y.	38	34	30	102
Every X is something that is not a Y.(*)	15	7	11	33
Everything that is not an X is a Y.	41	45	39	125
TOTAL	120	108	110	338

Table 6.13: Distribution of correct answers on three verbalisations for the deduction rule 'Top-DisCls' (rule 53 in Table 6.3); X and Y are class names, statements marked with * are those following from the premises, and those without a mark do not follow from the premises

Inference	**Verbalisation 1** Everything is a Y. No X is a Y.	**Verbalisation 2** Everything is a Y. Nothing is both an X and a Y.	**verbalisation 3** Everything is a Y. No Y is an X.	**TOTAL**
Nothing is an X.(*)	36	33	35	104
Everything is an X.	45	43	43	131
Every X is something that is not a Y.(*)	31	29	32	92
Everything that is not a Y is an X.(*)	20	6	18	44
TOTAL	132	111	128	371

6.9 (rule 55), 6.10 (rule 56), and 6.11 (rule 57).

Rules 52 and 53 have three candidate verbalisations. Friedman tests showed that there were no reliable differences in subjects' performance between the verbalisations of rule 52 (χ^2=4.84, df=2, p>0.05), but there were statistically reliable differences in subjects' performance between those of rule 53 (χ^2=10.16, df=2, p<0.01). Follow-up pairwise comparisons using a Wilcoxon Signed Ranks test for rule 53 showed that there were differences in subjects' performance between the verbalisations 1 and 2 (Z=2.91, p=0.004), 2 and 3 (Z=2.83, p=0.005), and verbalisation 2 was the worst. This means that the verbalisations 1 and 3 were equally good for this rule. The absolute frequencies of correct answers of these rules are summarised in Tables 6.12 (rule 52) and 6.13 (rule 53).

The Best Verbalisation for $X \equiv Y$

Table 6.14: Distribution of correct answers on two verbalisations of the axiom $X \equiv Y$ through all test questions of rules 49, 50, and 51 in Table 6.3, with X and Y are any class names

Deduction Rule	**Verbalisation 1** An X is anything that is a Y.	**Verbalisation 2** Every X is a Y; every Y is an X.	**TOTAL**
Rule 49	140	177	317
Rule 50	139	151	290
Rule 51	140	152	292
TOTAL	419	480	899

Table 6.15: Distribution of correct answers on two verbalisations of the axiom $\mathbf{Dis}(X, Y)$ through all test questions of rules 52, 53, 54, and 55 in Table 6.3, with X and Y are any class names

Deduction Rule	**Verbalisation 1** No X is a Y.	**Verbalisation 2** Nothing is both an X and a Y.	**TOTAL**
Rule 52	120	110	230
Rule 53	132	111	243
Rule 54	128	122	250
Rule 55	108	106	214
TOTAL	488	449	937

To identify the best among the two verbalisations "An X is anything that is a Y" and "Every X is a Y; every Y is an X" for the axiom $X \equiv Y$, we compared the absolute frequencies of correct answers through all test questions of rules 49-51. These data are summarised in Table 6.14. A Wilcoxon Signed Ranks test revealed a statistically reliable difference in subjects' performance between the two verbalisations ($Z=3.66$, $p<0.0001$), and the second wording was the best.

The Best Verbalisation for $\mathbf{Dis}(X, Y)$

Similarly, to identify the best among the two verbalisations "No X is a Y" and "Nothing is both an X and a Y" for the axiom $\mathbf{Dis}(X, Y)$, we compared the absolute frequencies of correct answers through all test questions of rules 52-55. These data are summarised in Table 6.15. A Wilcoxon Signed Ranks test revealed a statistically reliable difference in subjects' performance between the two verbalisations ($Z=2.67$, $p=0.008$), and the first wording was the best.

The Best Premise Ordering for $\mathbf{Dom}(R_o, X)$ and $\mathbf{Rng}(R_o, X)$

In rules 56 and 57, the main source of the multiplicity of verbalisations lies in the ordering of premises, and the axioms $\mathbf{Dom}(R_o, X)$ and $\mathbf{Rng}(R_o, X)$ function in the same

Table 6.16: Distribution of correct answers on two premise orderings of the axiom $\mathbf{Dom}(R_o, X)$ (or $\mathbf{Rng}(R_o, X)$) through all test questions of rules 56 and 57 in Table 6.3, with X is any class name, and R_o is any object property name

Deduction Rule	Ordering 1 $\sqsubseteq$-Dom/Rng	Ordering 2 Dom/Rng-$\sqsubseteq$	TOTAL
Rule 56	130	126	256
Rule 57	127	121	248
TOTAL	257	247	504

way. Therefore, to identify the best ordering of the axiom $\mathbf{Dom}(R_o, X)$ (or $\mathbf{Rng}(R_o, X)$) in these rules, we compared the absolute frequencies of correct answers through all of their test questions. These data are summarised in Table 6.16. However, a Wilcoxon Signed Ranks test showed no reliable differences in subjects' performance between the two orderings (Z=0.97, p=0.33).

6.5 Conclusions and Future Work

We have initiated research on the identification of the most understandable verbalisation for a deduction rule (or a single-step inference) in OWL. We have conducted a thorough examination of theoretical insights from the psychology of reasoning and a novel empirical study on candidate verbalisations for our deduction rules, and shown that for most of our rules the participants perform equally well on different candidate verbalisations. The study also shows that overall the verbalisation "Every X is a Y; every Y is an X" is the best for explaining the axiom $X \equiv Y$ (probably because it explicitly shows both subsumption relationships between two classes), the verbalisation "No X is a Y" is the best for explaining the axiom $\mathbf{Dis}(X, Y)$ (probably because it is concise), finally the axioms $\mathbf{Dom}(R_o, X)$ and $\mathbf{Rng}(R_o, X)$ can be presented first or last in a verbalisation for a rule (so, we choose to present them last—e.g., we select the verbalisation "Every X R_d a D_r. Anything that R_d some value is a Y. Therefore, Every X is a Y" for rule 56 in Table 6.3).

Part of the future work is to find better ways to verbalise property names (both object and data properties) as well as verbalise axioms that contain one or more property names. This would help to improve the readability of verbalisations for individual axioms in a rule. Additionally, we will investigate the elimination of obvious statements—such as ""X owns Y" means the same as "Y is owned by X""—from the verbalisations of our rules.

Chapter 7

Understandability of OWL Inferences

This chapter addresses the problem of assessing the understandability of inferences in OWL. Section 7.1 discusses earlier work related to this problem. Section 7.2 describes a method of empirically measuring the understandability of our deduction rules, which are in fact single-step OWL inferences. Based on this measurement, Section 7.3 then describes a model for predicting the understandability of an entire proof tree, which is in fact a multi-step OWL inference. An evaluation of this model is reported in Section 7.4, which confirms that our model works relatively well in detecting differences in understandability of two-step OWL inferences.

7.1 Related Work

Several support tools have been proposed to help ontology developers to identify the causes of class unsatisfiability [KPSH05], and to rewrite potentially problematic axioms [LSPV08]. Two studies have been conducted [KPSH05, LSPV08] to evaluate how developers debug ontologies with and without the tools. However, these studies focus on how people with a good understanding of OWL perform debugging, but not on how well they understand

OWL inferences.

In a deduction rule, the conclusion can be viewed as an entailment, and the premises can be viewed as a justification. Horridge et al. have proposed a model for measuring the cognitive difficulty of a justification [HBPS11, Hor11]. In this model, they provide a list of *components*, each of which has an associated weight. For a given justification, the model checks for all appearances of these components, sums the weighted numbers of occurrences of the components, and outputs the result as the justification's difficulty score. The choice of the components and their weights is based on the authors' observations from an exploratory study [HPS09b, Hor11] and their intuitions. Most of the proposed components are based on the syntactic analysis of justifications such as the number and types of premises in a justification, and these syntax-based components are mostly assigned a high weight. There are also several components for revealing difficult phenomena such as the trivial satisfaction of universal restriction in OWL[1]; however, the weights of these components are often low and are chosen intuitively. Therefore, this model predicts the difficulty of a justification in a manner that is biassed towards its structural complexity rather than its cognitive difficulty.

An empirical study has been conducted to evaluate how well the above-mentioned model predicts the difficulty of justifications. In this study, the authors create a deduction problem, presented in Manchester OWL Syntax [HDG+06, HPS08c] with alpha-numeric characters as class, property, and individual names, for testing a justification. In each problem, a justification and its entailment are given, and subjects are asked whether the justification implies the entailment. A weakness of this study is that positive response bias is not controlled—i.e., if subjects had a positive response bias, they would have answered most questions correctly, regardless of the difficulty of the questions. Additionally, this study tests the model based only on analysis of subjective understanding (but not analysis of objective understanding)[2].

The above-mentioned complexity model and evaluation study are, in fact, inspired by those of Newstead et al. [NBH+06], which are proposed for measuring the difficulty of "Analytical Reasoning" (AR) problems in Graduate Record Examination (GRE) tests. An AR problem is a deductive reasoning problem in which an initial scenario is given

[1] That is, if $\langle x, y\rangle \notin R^{\mathcal{I}}$ for all $y \in \Delta^{\mathcal{I}}$ then $x \in (\forall R.C)^{\mathcal{I}}$.

[2] As will be discussed shortly, our evaluation study for the understandability model controlled for positive response bias and relied on analysis of both subjective and objective understanding.

Figure 7.1: The test problem for the deduction rule 'ObjDom-ObjAll', which yields $\top \sqsubseteq X$ from $\mathbf{Dom}(R_o, X)$ and $\forall R_o.\bot \sqsubseteq X$

along with a number of constraints called *rules*, and the examinee is asked to determine a possible solution for the problem among five choices. Like Horridge et al., Newstead et al. identify a set of difficulty factors and their weights through an intensive pilot study, and build a preliminary difficulty model based on these factors and weights. After that, a series of large-scale studies are conducted to validate as well as adjust the model. Leaving aside the fact that these reasoning problems are different from OWL inferences, a strength of this work is that response bias of all types is successfully controlled. However, in both Newstead et al.'s and Horridge et al.'s work, there is no clear explanation of how weights are assigned, suggesting that the choice might have been based partly on intuition.

7.2 Measuring the Understandability of Deduction Rules

7.2.1 Materials

To measure the understandability of a rule, a deduction problem was devised in which premises of the rule were given in English, replacing class, individual, and property variables by fictional nouns, proper nouns, and verbs so that the reader would not be biassed by domain knowledge, and the subjects were asked whether the entailment of the rule followed from the premises. The correct answer was always "Follows".

As in our previous study (described in Section 6.4), we used non-entailment and trivial control questions to control positive response bias, as opposed to test questions. They either made statements about objects not mentioned in the set of consistent premises (in

which case, trivially, the correct answer was "Does not Follow"), or repeated one of the premises (in which case, also trivially, the correct answer was "Follows"). Each problem consisted of premises followed by two questions, one test question and one control. For half the problems, the correct answers were "Follows" and "Follows"; for the other half "Follows" and "Does not Follow", with question order varying so that the test question sometimes preceded the control, and sometimes followed it. The understandability of rule 'ObjDom-ObjAll' (rule 17 in Table 7.4), for example, was measured using data gathered from the deduction problem in Figure 7.1, with the entailment as the second question.

7.2.2 Method

The study was conducted on the crowdsourcing service CrowdFlower. We used the same set-up as in our previous study (described in Section 6.4). In this study we specified to collect 50 responses per problem, but since some subjects gave up part-way through, the number of subjects was over 100.

Of the 57 deduction rules collected, 51 rules were measured. Since the main inferences of six remaining rules were similar to those of the measured rules—specifically, both 'DatSom-DatRng' and 'DatMin-DatRng yielded an unsatisfiability based on the effect of the data property range axiom; both 'ObjMin-ObjFun' and 'DatMin-DatFun' yielded an unsatisfiability based on the functionality of a property; both 'ObjSom-Bot-1' and 'ObjSom-Bot-2' yielded an unsatisfiability based on the unsatisfiability of an existential restriction's filler; both 'ObjSom-ObjMin' and 'DatSom-DatMin' yielded a subsumption based on the semantic equivalence between $\exists R$ and $\geq 1R$ in object and data properties; both 'ObjSom-ObjDom' and 'DatSom-DatDom' yielded a subsumption based on the common effect of object and data property domain axioms; finally, both 'Top' and 'Bot' yielded a tautological subsumption based on the $\top$ and $\bot$ classes—their understandability was derived from the associated measured rules.

7.2.3 Control Questions and Response Bias

As in our previous study (described in Section 6.4), we checked subjects' performance on all control questions to confirm whether subjects were taking the survey seriously. Figure 7.2 shows that for over 100 participants, all answered around 75% or more of the control

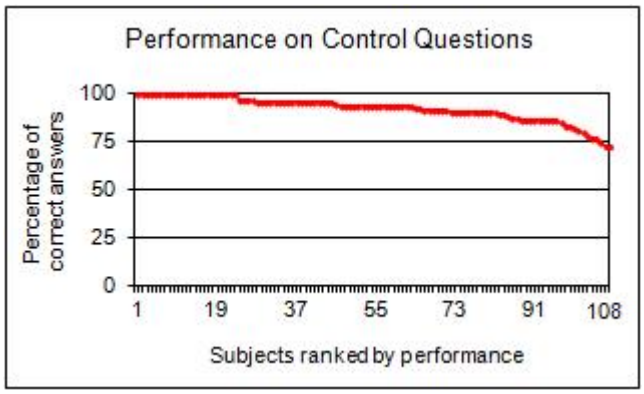


Figure 7.2: Subjects' performance on the control questions, sorted decreasingly

Table 7.1: Distribution of subjects' responses—i.e., "Follows" (+F) and "Does not Follow" (−F)—according to their correctness—i.e., "Correct" (+C) and "Incorrect" (−C)

	+F	**−F**	**TOTAL**
$+C$	2705	1195	3900
−C	118	912	1030
TOTAL	2823	2107	4930

questions correctly, suggesting that they were performing the task seriously. Among these subjects, only 2 claimed familiarity with OWL, 44 reported no familiarity, and the others did not specify (this question was optional).

Table 7.1 shows the absolute frequencies of the responses "Follows" (+F) and "Does not follow" (−F) for all questions in the study—control as well as test. It also subdivides these frequencies according to whether the answer was correct (+C) or incorrect (−C).

Recalling that for half the problems the correct answers were +F+F, while for half they are +F−F, the percentage of +F answers for a subject that always answered correctly would be 75%. If subjects had a positive response bias we would expect an overall rate higher than this, but in fact we obtained 2823/4930 or 57.3%, suggesting little or no positive response bias.

Looking at the distribution of incorrect answers, we can also ask whether subjects erred through being too ready to accept invalid conclusions (+F−C), or too willing to reject conclusions that were in reality valid (−F−C). The table shows a clear tendency towards the latter, with 912 responses in −F−C compared with an expected value of 440 (1030*2107/4930) calculated from the overall frequencies. In other words, subjects were more likely to err by rejecting a valid conclusion than by accepting an invalid one, a finding confirmed statistically by the extremely significant association between response (±F)

and correctness (±C) on a 2×2 chi-square test ($\chi^2 = 1116.3$, df = 1, $p < 0.0001$).

7.2.4 Facility Indexes

We use the proportion of correct answers for each test question as an index of understandability of the associated rule, which we call its *facility index* (FI). This index provides our best estimate of the probability that a person will understand the relevant inference step—that they will recognise that the conclusion follows from the premises—and accordingly ranges from 0.0 to 1.0.

Values of the facility indexes for the rules tested in this study are shown in Table 7.4, ordered from high values to low. In this table, rules 'SubCls-SubCls-1' and 'ObjDom-ObjAll' (used in the proof tree in Figure 1.1) are relatively easy, with facility indexes of 0.80 and 0.78. By contrast rule 'ObjAll', which yields statement (c) from axiom 1 in the example, is the most difficult, with a facility index of only 0.04, and hence, evidently a step in need of further elucidation (this problem is addressed in Chapter 8 in this thesis).

It can also be seen in Table 7.4 that for closely related rules, such as 'ObjSom-ObjMin', 'SubCls-SubCls-1' and even 'SubObj-SubObj', the facility indexes are quite close to each other (see also 'ObjDom-Bot' and 'ObjRng-Bot', 'DatVal-DatVal-DatFun' and 'ObjVal-ObjVal-DifInd-ObjFun', 'DatVal-DatRng' and 'DatSom-DatRng')—a result that confirms the reliability of the values. In general, the subjects found it difficult to understand inference steps that concluded an unsatisfiability (as in rules 'DatVal-DatVal-DatFun', 'ObjVal-ObjVal-DifInd-ObjFun', 'DatVal-DatRng', 'DatSom-DatRng' etc.), and those related to the unfamiliar behaviour of the universal restriction in OWL (as in rule 'ObjAll').

7.3 Predicting the Understandability of OWL Inferences

This section focuses on the understandability level of an entire proof tree, which can be viewed as a complex inference in OWL. When a tree has no lemma nodes, it corresponds to a single-step inference. Otherwise, it corresponds to a multi-step inference, like the one in Figure 1.1. We propose here a model which can predict the understandability of a multi-step inference based on the FIs of individual inference steps. Of course there is no fixed understandability for a given OWL inference as it depends on the readers' knowledge

of OWL as well as their deductive reasoning ability. For this reason, it is impossible to provide an accurate measurement of the understandability of an inference that is correct for most people. What we expect from this model is the ability to detect the *difference* in understandability between any two inferences. For example, if an inference is easier than another, then we expect that this model will be able to detect this.

To understand a complex inference consisting of multiple steps, it is essential to be able to understand each individual step within it. Recall that the FI of a rule provides our best estimate of the probability that a person will understand the relevant inference—i.e., that a person will recognise that the conclusion follows from the premises—thus it ranges from 0.00 to 1.00, and the higher this value, obviously, the easier. Given a proof tree with FIs assigned to each step, a natural method of combining indexes would be to multiply them, so computing the joint probability of all steps being followed—in other words, the *facility index* of the proof tree. As before, the higher this value, the easier the proof tree. According to this model, the understandability of the proof tree in Figure 1.1 would be 0.93*0.78*0.80*0.04*0.80 or 0.02, indicating that the proof tree is very difficult to understand. This prediction is supported by the claim from Horridge et al.'s study [HPS09b] that this inference is very difficult even for OWL experts.

7.4 Evaluation of the Model

In this section we report an experiment to evaluate the proposed model. This experiment focused on how well the model could detect differences in understandability between inferences. In this experiment, the use of *bins* for grouping inferences having close FIs was adapted from Horridge et al.'s study [HBPS11], but a different experimental protocol and materials were used. Additionally, not only objective but also subjective understanding of the participants was measured[3].

7.4.1 Materials

The experiment was carried out with 15 proof trees collected from the our corpus (described in Chapter 4). Each proof tree was assigned to an understandability bin on the

[3]All the materials and results of this study can found at `http://mcs.open.ac.uk/nlg/SWAT/ESWC2013.html`.

Table 7.2: The list of all tested inferences, their predicted FIs, and used deduction rules

ID	Tested Inference	FI	ID	Tested Inference	FI
1.1	$C_0 \equiv C_1$ $\wedge$ **Dom**(r_0, C_0) $\rightarrow$ **Dom**(r_0, C_1) *Rules: 'ObjDom-SubCls', 'ObjInt-2'*	**0.96**	2.1	**Rng**(r_0, C_1) $\wedge$ **Sym**(r_0) $\wedge$ $C_1 \sqsubseteq C_0$ $\rightarrow$ **Dom**(r_0, C_0) *Rules: 'ObjRng-ObjSym', 'ObjDom-SubCls'*	**0.74**
1.2	$C_0 \sqcup C_1 \sqsubseteq C_2$ $\wedge$ $C_0 \sqsubseteq C_3$ $\rightarrow$ $C_0 \sqsubseteq C_2 \sqcap C_3))$ *Rules: 'ObjUni-2', 'SubCls-SubCls-2'*	**0.90**	2.2	$C_0 \sqsubseteq C_1$ $\wedge$ $C_1 \sqsubseteq C_2$ $\wedge$ **Rng**(r_0, C_0) $\rightarrow$ **Rng**(r_0, C_2) *Rules: 'SubCls-SubCls-1', 'ObjRng-SubCls'*	**0.72**
1.3	$C_0 \sqsubseteq C_1 \sqcap C_2$ $\wedge$ **Rng**(r_0, C_0) $\rightarrow$ **Rng**(r_0, C_1) *Rules: 'ObjInt-2', 'ObjRng-SubCls'*	**0.86**	2.3	$C_1 \equiv C_2 \sqcup C3$ $\wedge$ $C_0 \sqsubseteq C_2$ $\rightarrow$ $C_0 \sqsubseteq C_1$ *Rules: 'ObjUni-1', 'SubCls-SubCls-1'*	**0.66**
3.1	$\neg C_1 \sqsubseteq C_2$ $\wedge$ $C_1 \sqsubseteq C_0$ $\wedge$ $C_2 \sqsubseteq C_0$ $\rightarrow$ $\top \sqsubseteq C_0$ *Rules: 'ObjCom-2', 'ObjUni-SubCls-SubCls'*	**0.53**	4.1	**Rng**(r_0, C_1) $\wedge$ **Invs**(r_1, r_0) $\wedge$ $C_0 \sqsubseteq \exists r_1.C_2$ $\rightarrow$ $C_0 \sqsubseteq C_1$ *Rules: 'ObjRng-ObjInv', 'ObjSom-ObjDom'*	**0.34**
3.2	$r_0 \sqsubseteq r_1$ $\wedge$ $r_1 \sqsubseteq r_2$ $\wedge$ **Dom**(r_2, C_0) $\rightarrow$ **Dom**(r_0, C_0) *Rules: 'SubObj-SubObj', 'ObjExt'*	**0.48**	4.2	$C_0 \sqsubseteq \exists r_0.C_2$ $\wedge$ **Rng**(r_0, C_1) $\wedge$ $DisCla(C_1, C_2)$ $\rightarrow$ $C_0 \sqsubseteq \bot$ *Rules: 'ObjSom-ObjRng', 'ObjInt-DisCls'*	**0.32**
3.3	$C_0 \sqsubseteq \geq 1 r_1.C_2$ $\wedge$ $r_1 \sqsubseteq r_0$ $\wedge$ $\exists r_0.C_2 \sqsubseteq C_1$ $\rightarrow$ $C_0 \sqsubseteq C_1$ *Rules: 'ObjSom-SubObj', 'ObjSom-ObjMin'*	**0.45**	4.3	$C_2 \sqsubseteq \forall r_0.C_1$ $\wedge$ **Invs**(r_0, r_1) $\wedge$ $C_0 \sqsubseteq \exists r_1.C_2$ $\rightarrow$ $C_0 \sqsubseteq C_1$ *Rules: 'ObjAll-ObjInv', 'SubCls-SubCls-1'*	**0.26**
5.1	**Fun**(d_0) $\wedge$ $C_0 \sqsubseteq \exists d_0.\{l_0\} \star DT_0))$ $\wedge$ $C_0 \sqsubseteq \exists d_0.\{l_1\} \star DT_0)), l_1 \neq l_0$ $\wedge$ $C_1 \sqsubseteq ObjMinCard(2, r_0, C_0))$ $\rightarrow$ $C_1 \sqsubseteq \bot$ *Rules: 'DatVal-DatVal-DatFun', 'ObjSom-Bot-1'*	**0.18**			
5.2	$C_1 \sqsubseteq existsr_0.(\exists d_0.l_0 \star DT_0)$ $\wedge$ **Rng**$(d_0, D_{t1}), DT_0 and D_{t1} are disjoint$ $\wedge$ $C_0 \sqsubseteq \exists r_1.C_1$ $\rightarrow$ $C_0 \sqsubseteq \bot$ *Rules: 'DatVal-DatRng', 'ObjSom-Bot-1'*	**0.09**			
5.3	$C_0 \equiv \forall r_0.C_1$ $\wedge$ **Dom**(r_0, C_0) $\rightarrow$ $\top \sqsubseteq C_0$ *Rules: 'ObjAll', 'ObjDom-ObjAll'*	**0.03**			

basis of the FI predicted by our model. For our purpose, a total of five understandability bins were constructed over the range from 0.00 to 1.00, each with an interval of 0.20[4]. For simplicity, only proof trees consisting of exactly *two* deduction rules (i.e., two-step inferences) were tested. The test proof trees were selected so that there would be *three* for each bin, and they would cover as many rules as possible. In fact, the test proof trees in this experiment included 25 of 51 rules from Table 7.4. The list of tested inferences, rules involved, and their predicted FIs is shown in Table 7.2.

For each proof tree, a *test problem* was devised in which the proof tree was given to the subjects in the form of a simple explanation in English, and the subjects were asked whether this explanation was correct. The subject was also asked to rank how difficult they found the question on a scale from 5 (very easy) to 1 (very difficult). When presenting the trees, fictional nouns and verbs were used so that the readers would not be biased by domain knowledge, and labels such as (a), (b) etc. were used to help them locate the statements quicker. As an example, the test problem for the test inference 5.3 in Table 7.2

[4]The ranges of the five bins were as follows: (B1) $0.80 < x \leq 1.00$, (B2) $0.60 < x \leq 0.80$, (B3) $0.40 < x \leq 0.60$, (B4) $0.20 < x \leq 0.40$, and (B5) $0 \leq x \leq 0.20$, respectively. B1 was the easiest, and B5 was the most difficult.

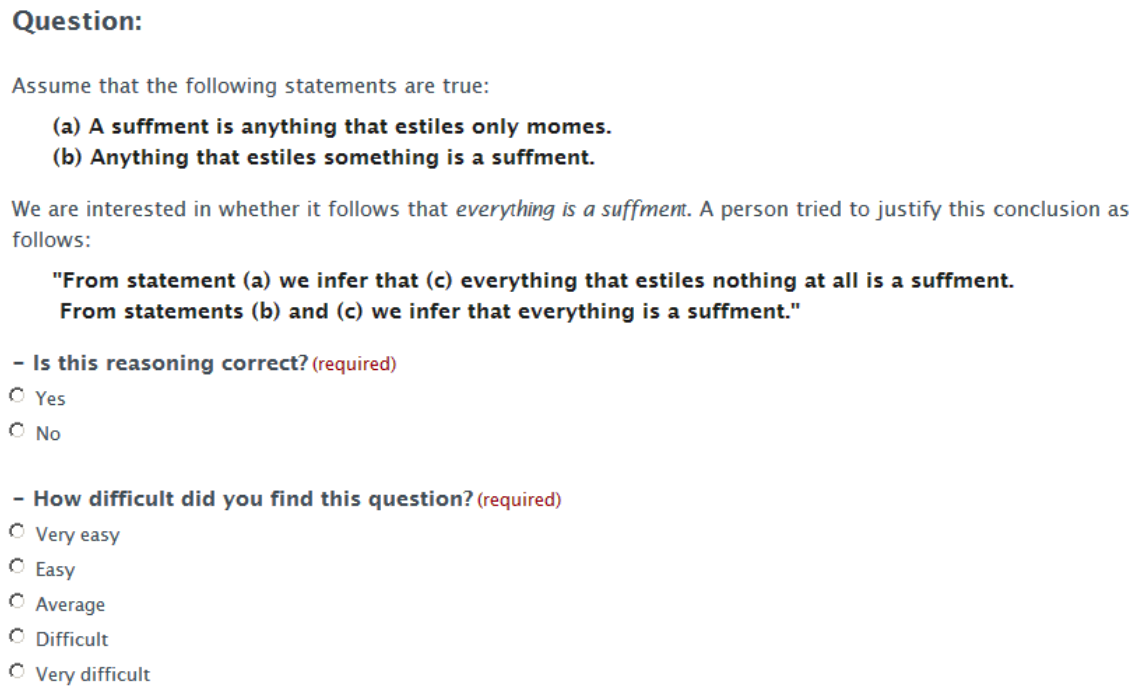


Figure 7.3: The test problem for the test inference 5.3 in Table 7.2 in which the FI of the proof tree is 0.03 (0.04 * 0.78)

is shown in Figure 7.3.

Since the correct answers to all test questions were "Yes", response bias was checked by including a number of *control problems*. In this study, control problems were designed to resemble the test problems, but obvious to subjects who did the test seriously. Two types of control problems were created: *non-entailment* and *trivial* problems. In a non-entailment problem, the test proof tree included a lemma or a conclusion about an object, a relationship, or both, that were not mentioned in the premises. The correct answer for non-entailment problems was "No", trivially. In order to create such problems, three possibilities for which the entailment was invalid were examined:

1. First inference step was invalid, second inference step was valid
2. First inference step was valid, second inference step was invalid
3. Both inference steps were invalid

Among the three cases, one would expect fewer mistakes for the third one since they had two opportunities to detect a mistake in the reasoning. Therefore, either the first or the second case was used in this study. In these cases, unrelated objects could not be introduced into a premise as this violated the assumption in a test problem that all given

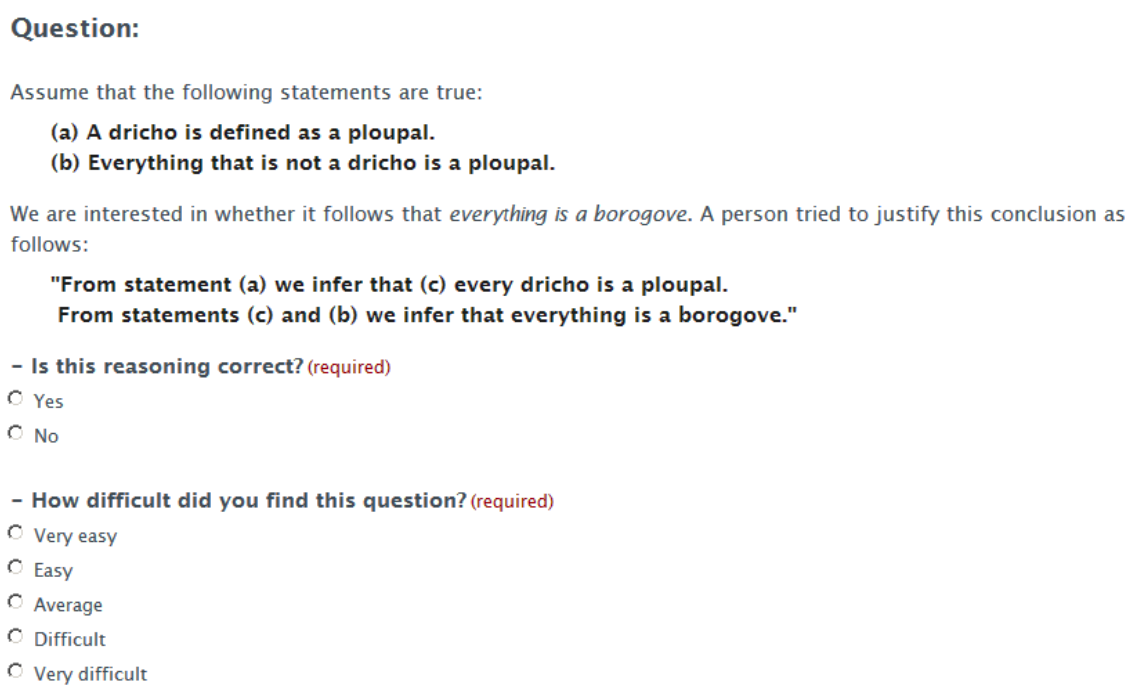


Figure 7.4: A non-entailment control problem for which the correct answer is "No"

premises were true; therefore, new objects were only introduced into the lemma in the first case or the entailment in the second case. An example non-entailment problem in this study is shown in Figure 7.4.

A trivial problem was one in which the test proof tree included only obviously correct inference steps, so the correct answer was, also trivially, "Yes". Making trivial problems was quite tricky in this study as repetitions of premises could not be merely used as before. This was because people might get confused about whether a statement explained an entailment if it merely repeated the entailment. Since people usually reasoned better with individuals than with general statements, inferences with individuals were used in trivial control problems. An example of such control problems is shown in Figure 7.5.

There were 15 test problems for which the correct answers were always positive. For balancing, 15 control problems were created, five of which having positive answers and the remaining problems having negative answers. This resulted in 20 positive and 10 negative problems—i.e., 67% positive vs. 33% negative.

7.4.2 Method

Like the previous ones, this study was conducted on the crowdsourcing service CrowdFlower with a similar set-up. We specified 80 responses per problem, but since we were

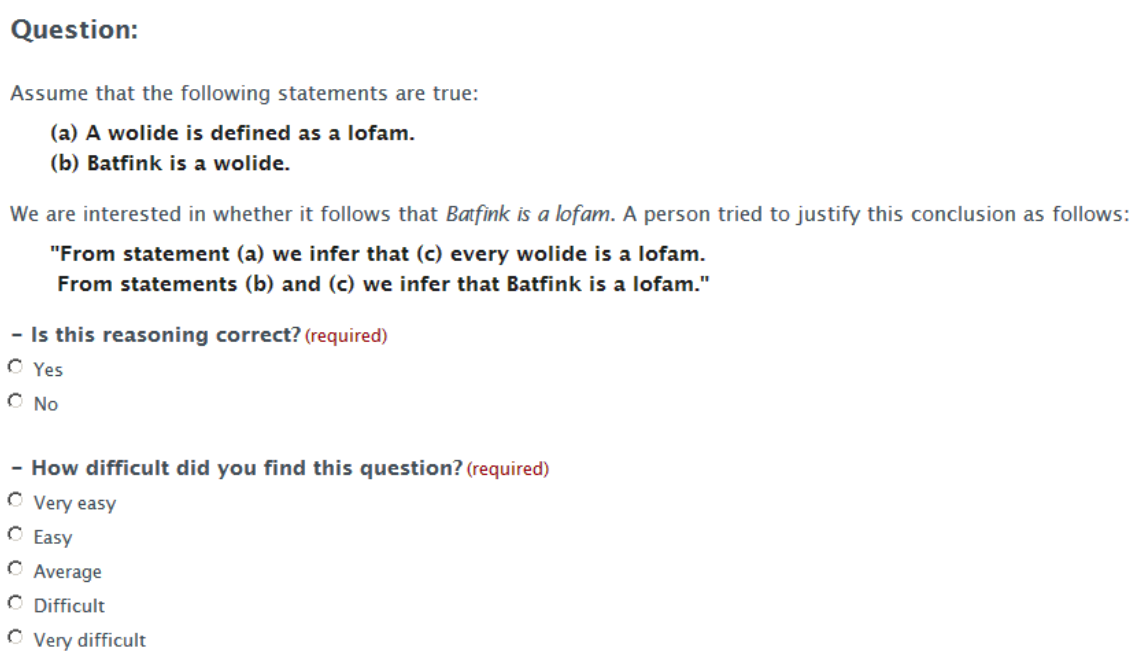


Figure 7.5: A trivial control problem for which the correct answer is "Yes"

only interested in responses in which all 30 problems were answered, only responses of 59 subjects were selected.

7.4.3 Control Questions and Response Bias

Figure 7.6 shows that among 59 participants, there were 7 who answered fewer than 70% of the control questions correctly, suggesting that they were not performing the test seriously; their results were accordingly discarded. Of the 52 subjects remaining, only one claimed familiarity with OWL, 45 reported no familiarity, and the others did not specify (this question was optional).

Table 7.3 shows the absolute frequencies of the subjects' responses "Yes" (+Y) and "No" (−Y) for all problems in the study—both control and test. It also subdivides these frequencies according to whether the response was correct (+C) or incorrect (−C). Thus for instance the cell +Y+C counts cases in which subjects answered "Yes" when this was the correct answer, while +Y−C counts cases in which they answered "Yes" when this was incorrect.

Recall that for 67% of the problems the correct answers were "Yes", and for all the remaining problems they were "No". If subjects had a positive response bias we would expect an overall rate much higher than 67%, but in fact we obtained 833/1556 or 54%,

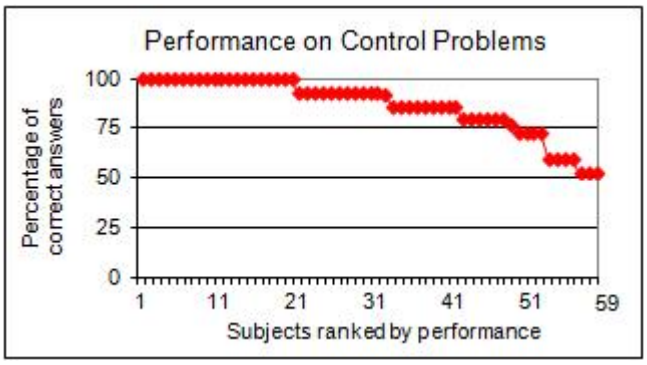


Figure 7.6: Subjects' performance on the control problems, sorted decreasingly

Table 7.3: Distribution of subjects' responses—"Yes" (+Y) and "No" (−Y)—according to their correctness—"Correct" (+C) and "Incorrect" (−C)

	+Y	**−Y**	**TOTAL**
+C	774	458	1232
−C	59	265	324
TOTAL	833	723	1556

suggesting no positive response bias.

As before, the distribution of incorrect answers in Table 7.3 shows that subjects were more likely to err by rejecting a valid conclusion than by accepting an invalid one (with 265 responses in −Y−C compared with an expected value of 324*723/1556=151 calculated from the overall frequencies), a finding confirmed statistically by the extremely significant association between response (±Y) and correctness (±C) on a 2×2 chi-square test (χ^2=205.3, df=1, p<0.0001).

7.4.4 Analysis of Objective Understanding

Figure 7.7 shows the relationship between the predicted FIs and the proportions of correct answers for test proof trees. The analysis indicated a statistically reliable relationship between the two values (r=0.88, p<0.0001) (Pearson's r correlation). For most test proof trees, the predicted FIs were lower than the actual proportions of correct answers. A possible explanation was that all of the control questions in this study were two-step inferences whereas those in the previous study were single-step inferences, and the use of more complex control questions in this study might have caused us to recruit better subjects than those of the previous one. However, for detecting differences in understandability of proof trees, the proposed model worked relatively well. Among the 15 test trees, there were 105 pairs on which difficulty comparisons could be made; of these, 93 comparisons

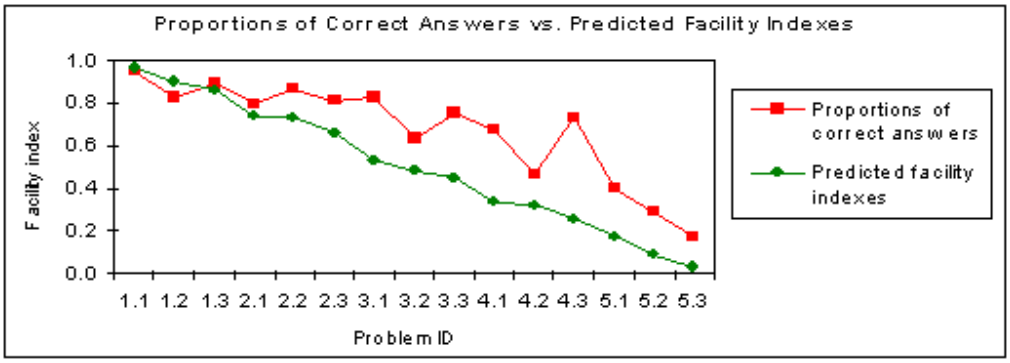


Figure 7.7: The predicted FIs vs. the proportions of correct answers

were ordered in difficulty as predicted (i.e., an accuracy of 89%).

We also tested how well our model could detect differences in understandability of proof trees by analysing the performance of the subjects by bins. For each subject, the number of correct answers for three test problems in each bin was counted, so obtaining a value of 0 to 3. A Friedman test was applied on the obtained values, which confirmed that there were statistically reliable differences in subjects' performance among the five bins (χ^2=108.95, df=4, p<0.0001). Follow-up pairwise comparisons using a Wilcoxon Signed Ranks test showed that there were differences in performance between any bin pair (p<0.05) except between 2 and 3. This might be because subjects found problems 3.1 and 3.3 easier than expected, thus reducing the difference between the two bins.

It is also clear from Figure 7.7 that there were exceptional cases for which the participants performed much better than we expected, such as proof trees 4.3, 4.1, 3.3, and 3.2. The changes of verbalisations used in this study might be the main reason for these exceptions. Proof trees 4.1 and 4.3 were the only two cases which included an $\mathbf{Invs}(r1, r0)$ axiom. In the previous study, this axiom was verbalised as "X r0 Y if and only if Y r1 X" (in rules 'ObjRng-ObjInv' and 'ObjAll-ObjInv'). The FIs we measured for these rules when using this verbalisation were 0.40 and 0.32 respectively. In this study, the axiom was verbalised as ""X r0 Y" means the same as "Y r1 X"", which was less technical, for testing trees 4.1 and 4.3; this might explain why the participants performed better on these trees than we expected. The proportions of correct answers for trees 4.1 and 4.3 were 0.67 and 0.73.

Similarly, proof trees 3.2 and 3.3 were the only two cases which included $r0 \sqsubseteq r1$ axioms. In the previous study, this axiom was verbalised as "The property r1 is a sub-property of r0" (in rules 'ObjDom-SubObj' and 'ObjSom-SubObj'). The FIs we measured for these

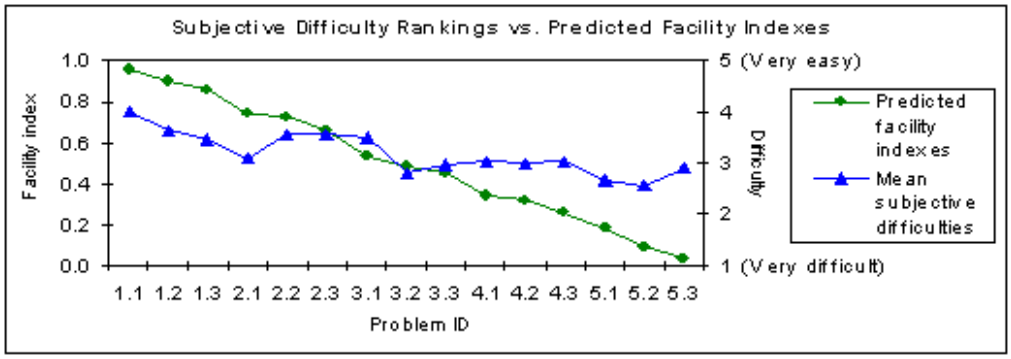


Figure 7.8: The predicted FIs vs. the mean subjective difficulty ratings

rules when using this verbalisation were 0.61 and 0.55. In the present study, we used the less technical verbalisation "If X r0 Y then X r1 Y" for this axiom, which might again explain why performance on these trees was better than we expected. The proportions of correct answers for trees 3.2 and 3.3 were 0.63 and 0.75.

7.4.5 Analysis of Subjective Understanding

Figure 7.8 plots the predicted FIs for test proof trees against the mean difficulty ratings (ranging from 1, very difficult, to 5, very easy) reported by subjects. The correlation between FIs and difficulty ratings was high (r=0.85) and significant ($p<0.0001$) (Pearson's r correlation).

As in the analysis of objective understanding, we tested how our model can detected differences in understandability of proof trees by analysing difficulty rankings by bins. For each subject, the mean value of difficulty rankings for the three questions of each bin was computed, and so obtained a value of 0 to 5. A Friedman test was applied on the obtained values, which confirmed that there were statistically reliable differences in difficulty ranking between the five bins (χ^2=88.66, df=4, $p<0.0001$). Follow-up pairwise comparisons using a Wilcoxon Signed Ranks test showed that there were statistically reliable differences in difficulty ranking between any bin pair ($p<0.05$) except between bins 3 and 4, for which the results might have been affected (as explained in the previous subsection) by the more accessible verbalisations used in the present study for the proof trees 3.2, 3.3, 4.1, and 4.3. Proof tree 5.3 was an exception as it was ranked as easier than 5.2 while our model predicted the opposite direction. This prediction was supported by the analysis of objective understanding presented previously. This result suggested a failure

in understanding this proof tree—i.e., the subjects thought that they had understood the inference correctly but actually they had not.

Table 7.4: Deduction rules and their facility indexes (FI), with 'CA' means the absolute number of correct answers and 'S' means the absolute number of subjects. For brevity, the names of OWL functors are abbreviated.

ID	Rule Name	Test Pattern	Deduction Problem	CA	S	FI
1	EquCls	$X \equiv Y$	A hiatea is defined as a milvorn.	49	49	1.00
		$\rightarrow X \sqsubseteq Y$	→Every hiatea is a milvorn.			
2	ObjInt-2	$X \sqsubseteq Y \sqcap Z$	Every ormyrr is both a gargoyle and a harpy.	47	49	0.96
		$\rightarrow X \sqsubseteq Y$	→Every ormyrr is a gargoyle.			
3	ObjDom-SubCls	$\mathbf{Dom}(R_o, X)$	Anything that has a supernatural ability is a bulette.	45	47	0.96
		$\wedge X \sqsubseteq Y$	Every bulette is a manticore.			
		$\rightarrow \mathbf{Dom}(R_o, Y)$	→Anything that has a supernatural ability is a manticore.			
4	ObjUni-2	$Y \sqcup Z \sqsubseteq X$	Everything that is a volodni or a treant is a maradan.	44	46	0.96
		$\rightarrow Y \sqsubseteq X$	→Every volodni is a maradan.			
5	SubCls-SubCls-2	$X \sqsubseteq Y$	Every bullywug is a grippli.	45	48	0.94
		$\wedge X \sqsubseteq Z$	Every bullywug is a prismatic.			
		$\rightarrow X \sqsubseteq (Y \sqcap Z)$	→Every bullywug is both a grippli and a prismatic.			
6	Top	$\top \sqsubseteq X$	Everything is a kelpie.	43	46	0.93
		$\rightarrow Y \sqsubseteq X$	→Every person is a kelpie.			
7	ObjSom-ObjAll-2	$X \sqsubseteq \exists R_o.\top$	Every locathah eats something.	45	50	0.90
		$\wedge X \sqsubseteq \forall R_o.Y$	Every locathah eats only orogs.			
		$\rightarrow X \sqsubseteq \exists R_o.Y$	→Every locathah eats an orog.			
8	ObjRng-SubCls	$\mathbf{Rng}(R_o, X)$	Anything that something lives in is a tarrasque.	44	49	0.90
		$\wedge X \sqsubseteq Y$	Every tarrasque is a kraken.			
		$\rightarrow \mathbf{Rng}(R_o, Y)$	→Anything that something lives in is a kraken.			
9	ObjSom-ObjDom	$X \sqsubseteq \exists R_o.Z$	Anything that is a messenger of something is a landwyrm.	43	50	0.86
		$\wedge \mathbf{Dom}(R_o, Y)$	Every spellgaunt is a messenger of a gravorg.			
		$\rightarrow X \sqsubseteq Y$	→Every spellgaunt is a landwyrm.			
10	ObjUni-1	$X \equiv Y \sqcup Z$	Every cooshee is a peryton or a banderlog; everything that is a peryton or a banderlog is a cooshee.	41	50	0.82
		$\rightarrow Y \sqsubseteq X$	→Every peryton is a cooshee.			
11	ObjSom-ObjMin	$X \sqsubseteq \exists R_o.Y$	Every varag lives on a seaplane.	40	49	0.82
		$\wedge \geq 1 R_o.Y \sqsubseteq Z$	Everything that lives on at least one seaplane is an urophion.			
		$\rightarrow X \sqsubseteq Z$	→Every varag is an urophion.			
12	SubCls-SubCls-1	$X \sqsubseteq Y$	Every sivak is a draconian.	37	46	0.80
		$\wedge Y \sqsubseteq Z$	Every draconian is a guulvorg.			
		$\rightarrow X \sqsubseteq Z$	→Every sivak is a guulvorg.			
13	ObjCom-1	$X \sqsubseteq \neg X$	Every zezir is something that is not a zezir.	40	50	0.80

Continued on Next Page...

		$\rightarrow X \sqsubseteq \bot$	→Nothing is a zezir.			
14	SubObj-SubObj	$R_o \sqsubseteq S_o$	The property "is a kobold of" is a sub-property of "is a drow of".	41	52	0.79
		$\wedge S_o \sqsubseteq T_o$	The property "is a drow of" is a sub-property of "is a tiefling of".			
		$\rightarrow R_o \sqsubseteq T_o$	→The property "is a kobold of" is a sub-property of "is a tiefling of".			
15	ObjSom-SubCls	$X \sqsubseteq \exists R_o.Y$	Every phaerlin is father of a firbolg.	37	47	0.79
		$\wedge Y \sqsubseteq Z$	Every firbolg is a gnoll.			
		$\rightarrow X \sqsubseteq \exists R_o.Z$	→Every phaerlin is father of a gnoll.			
16	ObjInt-1	$X \equiv Y \sqcap Z$	A cyclops is anything that is both a troofer and a gathra.	37	47	0.79
		$\rightarrow X \sqsubseteq Y$	→Every cyclops is a troofer.			
17	ObjDom-ObjAll	$\mathbf{Dom}(R_o, X)$	Everything that has a worship leader is a fomorian.	39	50	0.78
		$\wedge \forall R_o.\bot \sqsubseteq X$	Everything that has no worship leader at all is a fomorian.			
		$\rightarrow \top \sqsubseteq X$	→Everything is a fomorian.			
18	ObjRng-ObjSym	$\mathbf{Rng}(R_o, X)$	Anything that something is an abrian of is a grolantor.	36	47	0.77
		$\wedge \mathbf{Sym}(R_o)$	X is an abrian of Y if and only if Y is an abrian of X.			
		$\rightarrow \mathbf{Dom}(R_o, X)$	→Anything that is a sibling of something is a grolantor.			
19	SubCls-ObjCom-2	$X \sqsubseteq Y$	Every oblivion moss is a vegepygmy.	36	47	0.77
		$\wedge \neg X \sqsubseteq Y$	Everything that is not an oblivion moss is a vegepygmy.			
		$\rightarrow \top \sqsubseteq Y$	→Everything is a vegepygmy.			
20	ObjDom-Bot	$\mathbf{Dom}(R_o, X)$	There does not exist anything that is a grimlock of something.	39	51	0.76
		$\wedge \neg X \sqsubseteq Y$	Everything that is not an oblivion moss is a vegepygmy.			
		$\rightarrow \top \sqsubseteq \forall R_o.\bot$	→Everything is not a grimlock.			
21	ObjRng-Bot	$\mathbf{Rng}(R_o, X)$	There does not exist anything that something has as a catter.	37	49	0.76
		$\wedge X \sqsubseteq \bot$	Everything that is not an oblivion moss is a vegepygmy.			
		$\rightarrow \top \sqsubseteq \forall R_o.\bot$	→Everything has no catter at all.			
22	DisCls-SubCls-SubCls	$\mathbf{Dis}(\mathbf{X}, \mathbf{Y})$	No plant is an animal.	35	46	0.76
		$\wedge U \sqsubseteq X$	Every kalamanthis is a plant.			
		$\wedge V \sqsubseteq Y$	Every tendriculos is an animal.			
		$\rightarrow \mathbf{Dis}(U, V)$	→No kalamanthis is a tendriculos.			
23	ObjSom-ObjSom-ObjTra	$X \sqsubseteq \exists R_o.Y$	Every dero is a tendriculos of a harpy.	38	51	0.75
		$\wedge Y \sqsubseteq \exists R_o.Z$	Every harpy is a tendriculos of a tasloi.			
		$\wedge \mathbf{Tra}(R_o)$	If X is a tendriculos of Y and Y is a tendriculos of Z then X is a tendriculos of Z.			
		$\rightarrow X \sqsubseteq \exists R_o.Z$	→Every dero is a tendriculos of a tasloi.			
24	ObjUni-SubCls-SubCls	$X \sqsubseteq (U \sqcup V)$	Every mongrelfolk is a nilbog or a norker.	35	48	0.73
		$\wedge U \sqsubseteq Z$	Every nilbog is a skulk.			
		$\wedge V \sqsubseteq Z$	Every norker is a skulk.			

Continued on Next Page...

		$\rightarrow X \sqsubseteq Z$	→Every mongrelfolk is a skulk.			
25	ObjCom-2	$\neg X \sqsubseteq Y$	Everything that is not a spriggan is an orog.	36	50	0.72
		$\rightarrow \top \sqsubseteq X \sqcup Y$	→Everything is a spriggan or an orog.			
26	ObjUni-SubCls	$X \sqsubseteq (Y \sqcup Z)$	Every merfolk is a lizardfolk or a kobold.	35	49	0.71
		$\wedge\ Y \sqsubseteq Z$	Every lizardfolk is a kobold.			
		$\rightarrow X \sqsubseteq Z$	→Every merfolk is a kobold.			
27	ObjSom-ObjAll-1	$\exists R_o.Y \sqsubseteq X$	Everything that supervises a worg is a stirge.	35	49	0.71
		$\wedge\ \forall R_o.\bot \sqsubseteq X$	Everything that supervises nothing at all is a stirge.			
		$\rightarrow \forall R_o.Y \sqsubseteq X$	→Everything that supervises only worgs is a stirge.			
28	ObjDom-ObjSym	$\mathbf{Dom}(R_o, X)$	Anything that is an obliviax of something is a kraken.	34	49	0.69
		$\wedge\ \mathbf{Sym}(R_o)$	X is an obliviax of Y if and only if Y is an obliviax of X.			
		$\rightarrow \mathbf{Rng}(R_o, X)$	→Anything that something is an obliviax of is a kraken.			
29	ObjSom-ObjTra	$X \sqsubseteq \exists R_o.(\exists R_o Y)$	Every draconian is a spriggan of something that is a spriggan of a shifter.	34	50	0.68
		$\wedge\ \mathbf{Tra}(R_o)$	If X is a spriggan of Y and Y is a spriggan of Z then X is a spriggan of Z.			
		$\rightarrow X \sqsubseteq \exists R_o.Y$	→Every draconian is a spriggan of a shifter.			
30	ObjSom-ObjRng	$X \sqsubseteq \exists R_o.Y$	Every mudmaw resembles a jermlaine.	32	50	0.64
		$\wedge\ \mathbf{Rng}(R_o, Z)$	Anything that something resembles is a corollax.			
		$\rightarrow X \sqsubseteq \exists R_o.(Y \sqcap Z)$	→Every mudmaw resembles something that is both a jermlaine and a corollax			
31	Top-DisCls	$\top \sqsubseteq Y$	Everything is a darfellan.	30	47	0.64
		$\wedge\ \mathbf{Dis}(X, Y)$	No grippli is a darfellan.			
		$\rightarrow X \sqsubseteq \bot$	→Nothing is a grippli.			
32	ObjExt	$X \sqsubseteq= n_1 R_o.Y,\ n_1 \geq n_2 \geq 0$	Every oaken defender has exactly two dry leaves.	29	46	0.63
		$\rightarrow X \sqsubseteq\geq n_2 R_o.Y$	→Every oaken defender has at least one dry leaf.			
33	ObjDom-SubObj	$\mathbf{Dom}(R_o, X)$	Anything that gyres something is a tiefling.	28	46	0.61
		$\wedge\ S_o \sqsubseteq R_o$	The property “raths” is a sub-property of “gyres”.			
		$\rightarrow \mathbf{Dom}(S_o, X)$	→Anything that raths something is a tiefling.			
34	SubCls-DisCls	$X \sqsubseteq Y$	Every aasimar is a sirine.	30	53	0.57
		$\wedge\ \mathbf{Dis}(X, Y)$	No aasimar is a sirine.			
		$\rightarrow X \sqsubseteq \bot$	→Nothing is an aasimar.			
35	DisCls-SubCls-SubCls	$\mathbf{Dis}(\mathbf{X}, \mathbf{Y})$	Every needleman is a basidirond.	27	48	0.56
		$\wedge\ U \sqsubseteq X$	Every needleman is a battlebriar.			
		$\wedge\ V \sqsubseteq Y$	No basidirond is a battlebriar.			
		$\rightarrow \mathbf{Dis}(U, V)$	→Nothing is a needleman.			
36	ObjTra-ObjInv	$\mathbf{Tra}(R_o)$	If X toves Y and Y toves Z then X toves Z.	27	49	0.55
		$\wedge\ \mathbf{Invs}(R_o, S_o)$	X toves Y if and only if Y is toved by X.			

Continued on Next Page...

		$\rightarrow$ **Tra**(S_o)	$\rightarrow$If X is toved by Y and Y is toved by Z then X is toved by Z.			
37	ObjSom-SubObj	$X \sqsubseteq \exists R_o.Y$	Every halfling is an ascomoid of a kenku.	28	51	0.55
		$\wedge\ R_o \sqsubseteq S_o$	The property "is an ascomoid of" is a sub-property of "is a basidirond of".			
		$\rightarrow X \sqsubseteq \exists S_o.Y$	$\rightarrow$Every halfling is a basidirond of a kenku.			
38	ObjRng-SubObj	**Rng**(R_o, X)	Anything that something brilligs is a girallon.	24	46	0.52
		$\wedge\ S_o \sqsubseteq R_o$	The property "gimbles" is a sub-property of "brilligs".			
		$\rightarrow$ **Rng**(S_o, X)	$\rightarrow$Anything that something gimbles is a girallon.			
39	SubCls-ObjCom-1	$X \sqsubseteq Y$	Every darkmantle is a gorgon.	25	49	0.51
		$\wedge\ X \sqsubseteq \neg Y$	Every darkmantle is not a gorgon.			
		$\rightarrow X \sqsubseteq \bot$	$\rightarrow$Nothing is a darkmantle.			
40	ObjInt-DisCls	$X \sqsubseteq \exists R_o.(Y_1 \sqcap \ldots \sqcap Y_m)$, $m \geq 2$	Every daemonfey is preceded by something that is both an axani and a phoera.	25	50	0.50
		$\wedge$ **Dis**$(Y_1, \ldots, Y_m[, \ldots])$	No axani is a phoera.			
		$\rightarrow X \sqsubseteq \bot$	$\rightarrow$Nothing is a daemonfey.			
41	ObjMin-ObjMax	$X \sqsubseteq\geq n_1 R_o.Y$	Every jermlaine possesses at least three things.	22	46	0.48
		$\wedge\ X \sqsubseteq\leq n_2 R_o.Y$, $0 \leq n_2 < n_1$	Every jermlaine possesses at most one thing.			
		$\rightarrow X \sqsubseteq \bot$	$\rightarrow$Nothing is a jermlaine.			
42	ObjSom-Bot-1	$X \sqsubseteq \exists R_o.Y$	Every tasloi has as owner an aasimar.	20	44	0.45
		$\wedge\ Y \sqsubseteq \bot$	Nothing is an aasimar.			
		$\rightarrow X \sqsubseteq \bot$	$\rightarrow$Nothing is a tasloi.			
43	ObjMin-ObjFun	$X \sqsubseteq\geq n R_o.Y$, $n > 1$	Everything has as ratings at most one value.	20	49	0.41
		$\wedge$ **Fun**$(\mathbf{R_o})$	Every buckawn has as ratings at least four integer values.			
		$\rightarrow X \sqsubseteq \bot$	$\rightarrow$Nothing is a buckawn.			
44	ObjRng-ObjInv	**Rng**(R_o, X)	Anything that something gimbles from is a terlen.	19	47	0.40
		$\wedge$ **Invs**(R_o, S_o)	X gimbles from Y if and only if Y gimbles into X.			
		$\rightarrow$ **Dom**(S_o, X)	$\rightarrow$Anything that gimbles into something is a terlen.			
45	DatVal-DatVal-DatFun	$X \sqsubseteq \exists R_d.\{l_0 \star D_{t0}\}$	Everything has as power level at most one value.	18	45	0.40
		$\wedge\ X \sqsubseteq \exists R_d.\{l_1 \star D_{t1}\}$, D_{t0} & D_{t1} are disjoint, or $l_0 \neq l_1$	Every sirine has as power level an integer value of 5.			
		$\wedge$ **Fun**$(\mathbf{R_d})$	Every sirine has as power level an integer value of 7.			
		$\rightarrow$SubClaOf(X,$\bot$)	$\rightarrow$Nothing is a sirine.			
46	ObjVal-ObjVal-DifInd-ObjFun	$X \sqsubseteq \exists R_o.\{i\}$	Everything worships at most one thing.	17	44	0.39
		$\wedge\ X \sqsubseteq \exists R_o.\{j\}$	Every selkie worships Ashur.			
		$\wedge$ **Dif**$(\mathbf{i}, \mathbf{j})$	Every selkie worships Enki.			
		$\wedge$ **Fun**$(\mathbf{R_o})$	Ashur and Enki are different individuals.			
		$\rightarrow X \sqsubseteq \bot$	$\rightarrow$Nothing is a selkie.			
47	ObjDom-ObjInv	**Dom**(R_o, X)	Anything that gimbles from something is an atomie.	18	48	0.38

Continued on Next Page...

		$\wedge$ **Invs**(R_o, S_o) $\rightarrow$ **Rng**(S_o, X)	X gimbles from Y if and only if Y gimbles into X. →Anything that something gimbles into is an atomie.			
48	ObjAll-ObjInv	$X \sqsubseteq \forall R_o.Y$ $\wedge$ **Invs**(R_o, S_o) $\rightarrow \exists S_o.X \sqsubseteq Y$	Every tabaxi toves from only lamias. X toves from Y if and only if Y toves into X. →Everything that toves into a tabaxi is a lamia.	16	50	0.32
49	DatVal-DatRng	$X \sqsubseteq \exists R_d.\{l \star D_t\})$ $\wedge$ **Rng**(R_d, D_r), D_t & D_r are disjoint $\rightarrow X \sqsubseteq \bot$	Any value that something has as dark-vision is an integer value. Every ettin makes friends with something that has as dark-vision string value of "three". String values are unconvertible to integer values in OWL. →Nothing is an ettin.	9	48	0.19
50	DatSom-DatRng	$X \sqsubseteq \exists R_d.D_{r0}$ $\wedge$ **Rng**(R_d, D_{r1}), D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	Any value that something has as life expectancy is an integer value. Every tiefling has as life expectancy a double value. Double values are unconvertible to integer values in OWL. →Nothing is a tiefling.	9	49	0.18
51	ObjAll	$X \equiv \forall R_o.Y$ $\rightarrow \forall R_o.\bot \sqsubseteq X$	A hiatea is anything that eats only lamias. →Everything that eats nothing at all is a hiatea.	2	49	0.04

7.5 Discussion and Conclusions

We have proposed a novel method for empirically measuring the understandability of single-step inferences in OWL, focusing on people with limited knowledge of OWL. The results indicate that overall the subjects found it difficult to understand inference steps that conclude an unsatisfiability.

We have also proposed a probabilistic model for predicting the understandability of a multi-step inference based on measurement of the understandability of single-step inferences. First the facility indexes of 51 single-step OWL inferences are measured in an empirical study, resulting in estimates of the probability that a person will understand the inference. Then by multiplying the indexes of individual inference steps, the joint probability of all steps being followed can be computed as the facility index of the associated multi-step inference. Our evaluation has confirmed that the proposed model works relatively well for two-step inferences in OWL.

When generating explanations, the model is applied to determine the most understandable among alternative inferences from a justification, as well as to sort explanations in order of decreasing understandability when multiple justifications are found. A prototype of this model as a plug-in of the SWAT ontology editing tool, and will be published soon at `http://mcs.open.ac.uk/nlg/SWAT/`.

The proposed model grounds facility indexes in a well-established probabilistic interpretation. This gives us confidence that the good performance of the model on two-step inferences will extend to n-step inferences for n>2. This has, however, to be balanced with the somewhat better performance of the theoretically less well-founded approach of taking the minimum, which for two-step inferences achieves an accuracy of 94%. Further work is needed to compare these models for inferences with more than two steps.

Leaving aside the way the proposed model has been used in the work presented here, we believe that both the facility indexes of OWL inferences and the method for obtaining them might be useful in alternative models or contexts. Additionally, the proposed model can be used by others to predict the understandability of different kinds of inferences. Therefore, they are worth reporting as a resource for other researchers.

Chapter 8

Strategy-Based Explanations for Hard Deduction Rules

This chapter describes the identification of difficult-to-understand deduction rules (Section 8.1), and the exploration of potential strategies for explaining them to ordinary users who often have limited knowledge of OWL and logic (Section 8.2). An empirical test to find out which strategy is most suitable for each difficult rule is then described (Section 8.3).

8.1 Identification of Hard Deduction Rules

We investigate whether effective explanation strategies can be found for the ten most difficult deduction rules in our set, as measured by their FIs (all of which are 0.45 or less). On inspection, it turns out that some rules among these ten are almost the same, differing only in whether they contain an object property or a data property (rules 45 and 46 in Table 7.4) or whether they contain $\exists R_d.\{l \star D_t\}$ or $\exists R_d.D_r$ (rules 49 and 50 in Table 7.4). Eliminating duplicates, we focus on the eight rules shown in Table 8.1 (sorted in order of increasing understandability).

Table 8.1: Difficult deduction rules (sorted in order of increasing understandability), and special strategies to explain them; the number of 'X' in a cell is the number of strategy-based explanations that need to be tested

ID	Name	Deduction Rule	Original Verbalisation	Strategy 1: Explicate axioms to cancel the presupposition	Strategy 2: Exemplify axioms to show the contradiction	Strategy 3: Paraphrase difficult axioms	Strategy 4: Contextualise Invs axioms
1	ObjAll	$X \equiv \forall R_o.Y$ $\rightarrow \forall R_o.\bot \sqsubseteq X$	An X is anything that R_o only Ys. $\rightarrow$Everything that R_o nothing at all is an X.	X			
2	DatSom-DatRng	$X \sqsubseteq \exists R_d.D_{r0}$ $\wedge$ **Rng**(R_d, D_{r1}), D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	Every X R_d a D_{r0}. Any value that something R_d is a D_{r1}. D_{r0} values are not D_{r1} values. $\rightarrow$Nothing is an X.	X X	X	X	
3	ObjAll-ObjInv	$X \sqsubseteq \forall R_o.Y$ $\wedge$ **Invs**(R_o, S_o) $\rightarrow \exists S_o.X \sqsubseteq Y$	Every X R_o only Ys. "X R_o Y" means the same as "Y S_o X". $\rightarrow$Everything that S_o an X is a Y.			X	X
4	ObjDom-ObjInv	**Dom**(R_o, X) $\wedge$ **Invs**(R_o, S_o) $\rightarrow$ **Rng**(S_o, X)	Anything that R_o something is an X. "X R_o Y" means the same as "Y S_o X". $\rightarrow$Anything that something R_o is an X.			X	X
5	ObjVal-ObjVal-DifInd-ObjFun	$X \sqsubseteq \exists R_o.\{i\}$ $\wedge X \sqsubseteq \exists R_o.\{j\}$ $\wedge$ **Dif**$(\mathbf{i}, \mathbf{j})$ $\wedge$ **Fun**$(\mathbf{R_o})$ $\rightarrow X \sqsubseteq \bot$	Every X R_o i. Every X R_o j. i and j are different individuals. Everything R_o at most one thing. $\rightarrow$Nothing is an X.	X X	X	X	
6	ObjRng-ObjInv	**Rng**(R_o, X) $\wedge$ **Invs**(R_o, S_o) $\rightarrow$ **Dom**(S_o, X)	Anything that something R_o is an X. "X R_o Y" means the same as "Y S_o X". $\rightarrow$Anything that S_o something is an X.			X	X
7	ObjMin-ObjFun	$X \sqsubseteq \geq nR_o.Y$, $n > 1$ $\wedge$ **Fun**$(\mathbf{R_o})$ $\rightarrow X \sqsubseteq \bot$	Every X R_d at least n D_rs. Everything R_d at most one value. $\rightarrow$Nothing is an X.	X X	X	X X	
8	ObjSom-Bot-1	$X \sqsubseteq \exists R_o.Y$ $\wedge Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$	Every X R_o a Y. Nothing is a Y. $\rightarrow$Nothing is an X.	X X	X		

8.2 Special Strategies for Explanations

We proposed four special strategies to make explanations for difficult rules become more understandable. These strategies originated from an informal study in which the author and some colleagues proposed enhanced explanations for these rules, after which we collected together the strategies into four groups.

Strategy 1: Explicate premises to cancel the presupposition

In 1, 2, 5, 7, and 8 in Table 8.1 (namely 'ObjAll', 'DatSom-DatRng', 'ObjVal-ObjVal-DifInd-ObjFun', 'ObjMin-ObjFun', and 'ObjSom-Bot-1'), the source of difficulty plausibly lies in *presuppositions* [Bea97] caused by the words 'only' and 'every'. In rule 1, the word 'only' may cause the readers to presuppose that an X is anything that R_o one or more things, and all of these things are Ys. In fact, the accurate meaning of rule is that an X is anything that either R_o nothing at all, or if it R_o one or more things then all of these things are Ys. The first part of the meaning (i.e., things that R_o nothing at all) is implicit in the word 'only'. To help the readers cancel this presupposition, we explicate the original axiom by transforming it into an equivalent axiom by using a disjunction operator: $X \equiv (\forall R_o.\bot) \sqcup (\forall R_o.Y)$. Hence, the new verbalisation for the axiom is "An X is anything that R_o nothing at all, or R_o only Ys".

In rules 2, 5, 7, and 8, the word 'every' in the verbalisations of $\sqsubseteq$ axioms like "Every X R_o a Y", "Every X R_d a D_{r0} value" etc. may cause the readers to presuppose that there exists at least an instance of X. This presupposition contradicts the conclusion that "Nothing is an X", so the readers may be confused whether rule is true. As before, we cancel this presupposition by transforming each $\sqsubseteq$ axiom into a new $\sqsubseteq$ axiom by using a disjunction operator—specifically, $X \sqsubseteq \exists R_o.Y$ is transformed into $X \sqsubseteq (\exists R_o.Y) \sqcup \bot$, $X \sqsubseteq \exists R_d.D_{r0}$ into $X \sqsubseteq (\exists R_d.D_{r0}) \sqcup \bot$, and so on. The new verbalisations would be "Every X R_o a Y, or there are no Xs at all" and "Every X R_d a D_{r0} value, or there are no Xs" etc.

Another way to cancel the presupposition caused by the word 'every' is to use if-then statements. Specifically, we explicate the axiom $X \sqsubseteq \exists R_o.Y$ as "If there are any Xs then they all R_o a Y", the axiom $X \sqsubseteq \exists R_d.D_{r0}$ as "If there are any Xs then they all R_d a D_{r0} value", and so on. This strategy, however, is not applicable for rule 1 as its only premise is an $\equiv$ axiom (but not a $\sqsubseteq$ axiom). In our empirical study, only the explication strategy

using a disjunction is tested for rule 1, but both of the explication strategies are tested for rules 2, 5, 7, and 8, as summarised in Table 8.1.

Strategy 2: Exemplify premises to show the contradiction

For rules that conclude a class is unsatisfiable, namely rules 2, 5, 7, and 8 in Table 8.1, another source of difficulty may be the visibility of the logical contradiction in the description of the class. To make it more visible to the readers, we propose a method that exemplifies the premises. Specifically, a named individual is assumed to be an instance of the class in the conclusion, and we show that the existence of this individual leads to a contradiction. For example, to explain the contradiction in rule 2, we propose the following template for an explanation:

> Suppose there is an X named Rover. It follows from the premises that:
> - Rover R_d a D_{r0},
> - Rover R_d only D_{r1}s, and
> - D_{r0}s are not D_{r1}s.
>
> The existence of an X such as Rover leads to a contradiction. Therefore, there are no Xs.

In this template, the statement "Rover R_d a D_{r0}" is inferred from the axiom "$X \sqsubseteq \exists R_d.D_{r0}$" ("Every X R_d a D_{r0} value") and the assumption, and "Rover R_d only D_{r1} values" from $\mathbf{Rng}(R_d, D_{r1})$ ("Any value that something R_d is a D_{r1}"). Similar templates for rules 5, 7, and 8 can be created in the same way.

Strategy 3: Paraphrase difficult axioms

Another source of difficulty of these rules may lies in the verbalisations of difficult axioms. We propose alternative verbalisations for several axioms, and compare them with the original ones to identify which ones are understood best by non-logicians. The list of possible difficult verbalisations and the new paraphrases we propose are listed in Table 8.2. These axioms occur once in rules 2-6, and twice in rule 7 in Table 8.1.

Strategy 4: Contextualise Invs axioms

Invs is a difficult axiom to reason with. For rules that contains an **Invs** axiom in their premises, namely rules 3, 4, and 6 in Table 8.1, we propose a method to contextualise this axiom in order to make it better connected to the remaining premises. Specifically, in rule 3 ('ObjAll-ObjInv') we rely on the first premise which says "Every X R_o only Ys"

Table 8.2: New paraphrases for (plausibly) difficult OWL axioms

OWL Axiom	**Original Verbalisation**	**Paraphrase**
Rng(R_d, D_r)	Any value that something R_d is a D_r.	Everything R_d only D_rs.
Invs(R_o, S_o)	"X R_o Y" means the same as "Y S_o X".	If something R_o something then the latter S_o the former; additionally, if something S_o something then the latter R_o the former.
Fun$(\mathbf{R_o})$	Everything R_o at most one thing.	Everything R_o either zero or one thing.
$X \sqsubseteq \geq nR_o.Y$	Every X R_d at least n D_rs.	Every X R_d n or more D_rs.

to contextualise the **Invs** axiom as follows:

> "X R_o Y" means the same as "Y S_o X", from which it follows that "an X R_o a Y" means the same as "a Y S_o an X".

Similarly, based on the first axiom in rule 4 ('ObjDom-ObjInv') which says "Anything that R_o something is an X", we explain the **Invs** axiom as follows:

> "X R_o Y" means the same as "Y S_o X", from which it follows that "an X R_o something" means the same as "something S_o an X".

8.3 Evaluation of the Strategies

The aim of this study is to check whether the explanations for difficult rules generated from the proposed strategies are more understandable than the original ones. The distribution of the strategy-based explanations to be tested in this study is summarised in Table 8.1. In this table, the number of explanations in a cell (i.e., the number of 'X'(s)) depends on whether the associated strategy is applicable for the associated rule, as well as the number of sub-strategies within this strategy.

8.3.1 Materials

For each strategy-based or original explanation for a rule, we derived a *deduction problem* in which both the premises and the entailment of the rule were given in the form of a simple explanation, and the subjects were asked whether the explanation was correct. The subjects were also asked to rank how difficult they found the question on a scale

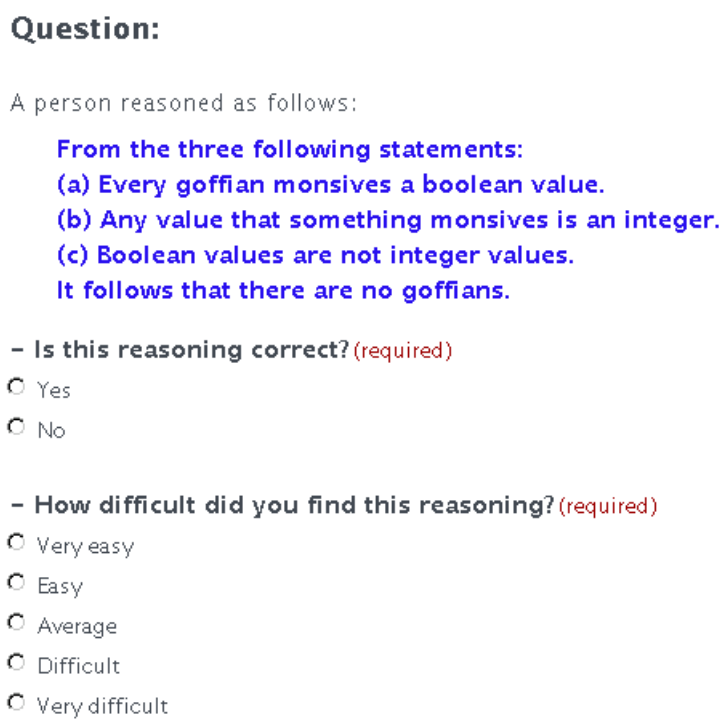


Figure 8.1: A test problem devised from the original explanation of the deduction rule 'DatSom-DatRng' (rule 2 in Table 8.1)

from 5 (very easy) to 1 (very difficult). In each explanation, both the premises and the entailment of the rule were given in English, replacing individual, class, and property variables by fictional names, nouns, and verbs so that the readers would not be biased by domain knowledge. Additionally, we used labels such as (a), (b) etc. to help subjects locate the statements quicker. For example, the test problems devised from the original explanation and the strategy-based explanation 1B (i.e., strategy 1 and using an if-then statement) for rule 2 in Table 8.1 are shown in Figure 8.1 and Figure 8.2, respectively.

As in the our previous study described in Section 7.4, we checked for response bias by including a number of *control problems*, as opposed to *test problems*, of two types: *non-entailment* and *trivial* problems. In a non-entailment problem, the conclusion of the explanation either included an object, a relationship, or both, that were not mentioned in the premises, or was invalid given that the premises were true (for easy inferences only). The correct answer for non-entailment problems was 'No', trivially. A trivial problem, in the other hand, was one in which the test explanation was an obviously correct inference, so the correct answer was, also trivially, 'Yes'. As before, we used inferences with individuals as trivial problems.

There were 23 strategy-based explanations plus 8 original explanations for 8 rules, so we

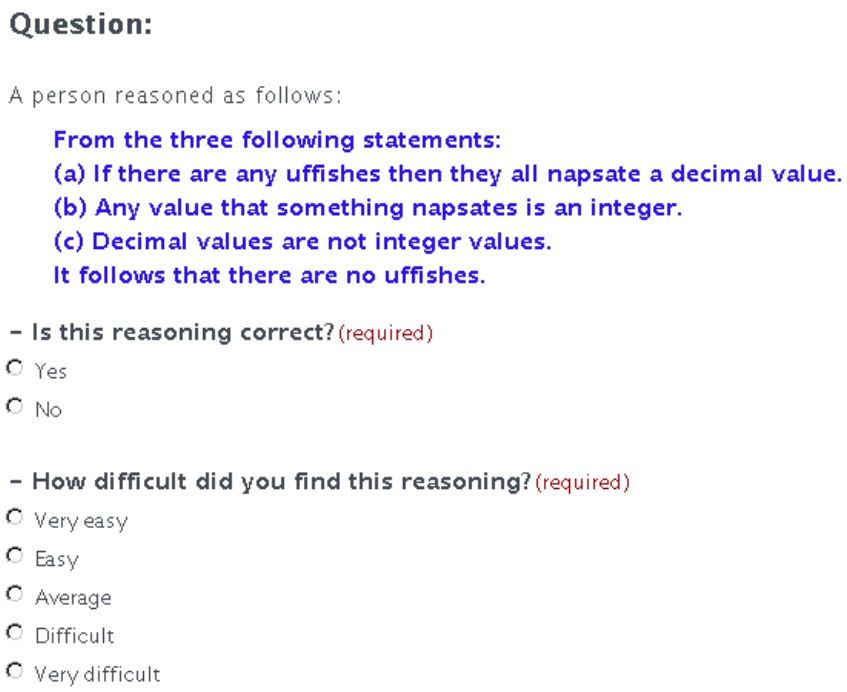


Figure 8.2: A test problem devised from strategy 1B (using an if-then statement) for the deduction rule 'DatSom-DatRng' (rule 2 in Table 8.1)

needed 31 test problems in total. Since this number was too large for a single study, we split into two smaller studies. In the first study, we tested 13 strategy-based explanations for the first two strategies (highlighted in Table 8.1), and 5 original explanations for the 5 associated rules (1, 2, 5, 7, and 8 in the table)—i.e., 18 test problems in total. We created 12 non-entailment and 5 trivial control problems for this study, resulting in 23/35 (66%) positive and 12/35 (34%) negative problems.

In the second study, we tested 10 remaining strategy-based explanations for the last two strategies, and 6 original explanations for the 6 associated rules (2-7 in Table 8.1)—i.e., 16 test problems in total. We also created 10 non-entailment and 2 trivial control problems, resulting in 18/28 (64%) positive and 10/28 (36%) negative problems.

8.3.2 Method

Both of the studies were conducted on CrowdFlower, using the same set-up as with the previous studies described in Chapter 7. We specified 70 responses per problem in each study, but since we were only interested in responses in which *all* test problems were answered, only responses of 55 subjects were selected in the first study, only responses of 54 subjects were selected in the second study, and none of these subjects participated in

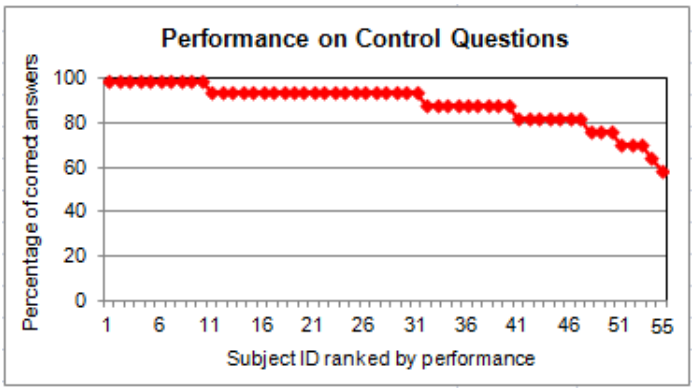


Figure 8.3: Subjects' performance on control questions in the first study, sorted decreasingly

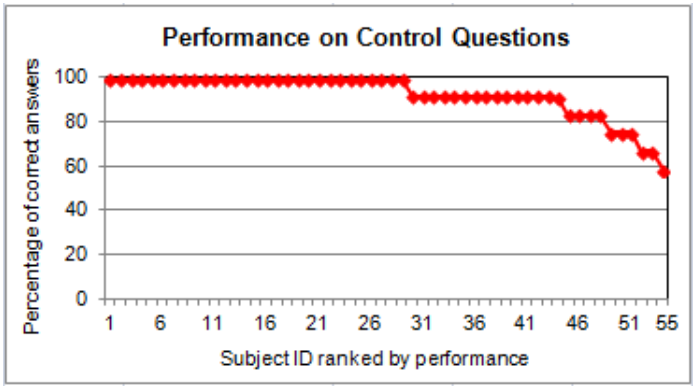


Figure 8.4: Subjects' performance on the control questions in the second study, sorted decreasingly

both of the studies.

8.3.3 Control Questions

Figures 8.3 shows that for the 55 subjects that participated in the first study, 53 answered 70% or more of the control questions correctly, suggesting that they were performing the task seriously. The responses of the other 2 subjects were discarded. Similarly, Figure 8.4 shows that 51 out of 54 subjects that participated in the second study answered 70% or more of the control questions correctly, and hence, only their responses were considered in our analysis.

8.3.4 Response Bias

For both studies, we obtained the same results as in previous studies for response bias. Tables 8.3 and 8.4 show the absolute frequencies of the subjects' responses 'Yes' (+Y) and

Table 8.3: Distribution of the subjects' responses—"Yes" (+Y) and "No" (−Y)—according to their correctness—"Correct" (+C) and "Incorrect" (−C)—in the first study

	+Y	**−Y**	**TOTAL**
+C	959	576	1535
−C	60	259	319
TOTAL	1019	835	1854

Table 8.4: Distribution of subjects' responses—"Yes" (+Y) and "No" (−Y)—according to their correctness—"Correct" (+C) and "Incorrect" (−C)—in the second study

	+Y	**−Y**	**TOTAL**
+C	723	473	1196
−C	32	194	226
TOTAL	755	667	1422

'No' (−Y) for all problems—both control and test—in both studies. They also subdivide these frequencies according to whether the response was correct (+C) or incorrect (−C).

Recall that for 66% of the problems in the first study and 64% of the problems in the second study the correct answers were 'Yes', and for all the remaining problems they were 'No'. In the first study, if subjects had a positive response bias we would expect an overall rate much higher than 66%, but in fact we obtained 1,019/1,854 or 55%, suggesting no positive response bias. Similarly, we would expect an overall rate much higher than 64% in the second study, but in fact we obtained 755/1,422 or 53%, suggesting no positive response bias in this study.

The distributions of incorrect answers in Tables 8.3 and 8.4 showed that subjects in both studies were more likely to err by rejecting a valid conclusion than by accepting an invalid one (with 259 responses in −Y−C compared with an expected value of 319*835/1,854=144 in the first study, and 194 responses in −Y−C compared with an expected value of 226*667/1,422=106 in the second study), findings confirmed statistically by the extremely significant associations between response (±Y) and correctness (±C) on 2×2 chi-square tests (χ^2=203.4, df=1, p<0.0001 for the first study; and χ^2=163.6, df=1, p<0.0001 for the second study).

Table 8.5: Results of the first study; for each test problem, the absolute frequency of correct answers and the associated percentage are provided, following by the mean of difficulty ratings (on a scale from 5 (very easy) to 1 (very difficult))

ID	Name	Deduction Rule	Original Explanation	Strategy 1A (Disjunction)	Strategy 1B (If-Then)	Strategy 2
1	ObjAll	$X \equiv \forall R_o.Y$ $\rightarrow \forall R_o.\bot \sqsubseteq$	7/53(13.2%) 3.8	35/53(66.0%) 3.8		
2	DatSom-DatRng	$X \sqsubseteq \exists R_d.D_{r0}$ $\wedge$ **Rng**(R_d, D_{r1}), D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	40/53(75.5%) 3.3	43/53(81.1%) 3.3	46/53(86.8%) 3.8	41/53(77.4%) 3.5
5	ObjVal-ObjVal-DifInd-ObjFun	$X \sqsubseteq \exists R_o.\{i\}$ $\wedge X \sqsubseteq \exists R_o.\{j\}$ $\wedge$ **Dif**$(\mathbf{i}, \mathbf{j})$ $\wedge$ **Fun**$(\mathbf{R_o})$ $\rightarrow X \sqsubseteq \bot$	44/53(83.0%) 4.0	45/53(84.9%) 3.6	42/53(79.2%) 3.8	40/53(75.5%) 3.7
7	ObjMin-ObjFun	$X \sqsubseteq\geq nR_o.Y$, $n > 1$ $\wedge$ **Fun**$(\mathbf{R_o})$ $\rightarrow X \sqsubseteq \bot$	40/53(75.5%) 3.6	42/53(79.2%) 3.8	43/53(81.1%) 4.0	40/53(75.5%) 3.5
8	ObjSom-Bot-1	$X \sqsubseteq \exists R_o.Y$ $\wedge Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$	38/53(71.7%) 4.0	51/53(96.2%) 4.0	43/53(81.1%) 4.0	42/53(79.2%) 3.7

8.3.5 Results of Study 1

Results of the first study are summarised in Table 8.5. Overall, the performance of the participants on the original explanations was much better than that measured in our previous study to measure the understandability of the rules (reported in Section 7.2). Perhaps the slightly better verbalisations as well as the presentation of a rule as a simple explanation in the present study helped the participants understand the relevant inferences more easily.

For rule 1 ('ObjAll') in Table 8.5, we found no reliable difference in *difficulty ranking* between the original explanation and explanation 1A (Wilcoxon Signed Ranks test, Z=0.03, p=0.97). However, there was a reliable difference in *subjects' performance* between the two explanations (McNemar's test, p<0.0001), and the subjects performed significantly better on explanation 1A than on the original explanation. This result suggested that most of the subjects thought that they had understood the original explanation but they actually had not, and that strategy 1A is useful for explaining this rule to ordinary people.

For rule 8 ('ObjSom-Bot-1') in Table 8.5, we found no reliable differences in *difficulty ranking* among the four cases (Friedman's test, χ^2=4.80, p=0.19). However, there wer statistically reliable differences in *subjects' performance* among the four cases (Cochran's Q test, Q(3)=13.35, p=0.004). Pairwise comparisons using a McNemar's test revealed

that there were reliable differences between the following pairs: the original explanation and 1A (p=0.002), 1A and 1B (p=0.008), and 1A and 2 (p=0.004); and 1A was the most understandable. This suggested that 1A would be the best strategy for this rule.

On the other hand, for rule 2 ('DatSom-DatRng') in Table 8.5, there were statistically reliable differences in *difficulty ranking* among the four cases (Friedman's test, χ^2=10.71, p=0.013). Pairwise comparisons using a Wilcoxon Signed Ranks test revealed that the difficulty ranking of explanation 1B was reliably greater than that of the original explanation (Z=3.26, p=0.001), 1A (Z=3.14, p=0.002), and 2 (Z=2.12, p=0.034), but there were no reliable differences between all other pairs.

For *subjects' performance*, we found no reliable differences among the four cases (Cochran's Q test, Q(3)=3.07, p=0.38). This suggested that the participants perceived strategy-based explanation 1B as the easiest one among the four cases, even though it did not really facilitate their understanding. Similar results were found for rules 5 ('ObjVal-ObjVal-DifInd-ObjFun') and 7 ('ObjMin-ObjFun') in Table 8.5.

In conclusion, we chose strategy 1A—that is, explicating the premises by using disjunctions to cancel the presuppositions—to generate explanations for rules 1 and 8. For rules 2, 5, and 7, although the participants perceived strategy 1B as easier than the others, no improvement in performance were found. Therefore, we also chose strategy 1A for these rules in order to generate explanations that were consistent with those for rules 1 and 8.

8.3.6 Results of Study 2

Results of the second study are summarised in Table 8.6. As in the first study, the performance of the participants on the original explanations was better than that measured in our previous study (reported in Section 7.2), and this might be caused by the use of slightly better verbalisations as well as the presentation of a rule as a simple explanation in this study.

For rules 3, 4, and 6, we tested the application of the two strategies 3 and 4 on the common axiom $\mathbf{Invs}(R_o, S_o)$ in their premises. Cochran's Q tests revealed no reliable differences in *subjects' performance* among the three cases in each rule (Q(2)=0.32, p=0.85 for rule 3; Q(2)=1.90, p=0.39 for rule 4; and Q(2)=4.52, p=0.10 for rule 6). However, the comparison of *subjects' performance* by strategies (original, 3, and 4) across the three rules (3, 4, and 6)

Table 8.6: Results of the second study; for each test problem, the absolute frequency of correct answers and the associated percentage are provided, following by the mean of difficulty rankings (on a scale from 5 (very easy) to 1 (very difficult))

ID	Name	Deduction Rule	Original Explanation	Strategy 3	Strategy 4
2	DatSom-DatRng	$X \sqsubseteq \exists R_d.D_{r0}$ $\wedge$ $\mathbf{Rng}(R_d, D_{r1})$, D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	39/51(76.5%) 3.4	40/51(78.4%) 3.3	
3	ObjAll-ObjInv	$X \sqsubseteq \forall R_o.Y$ $\wedge$ $\mathbf{Invs}(R_o, S_o)$ $\rightarrow \exists S_o.X \sqsubseteq Y$	43/51(84.3%) 3.5	42/51(82.4%) 3.5	44/51(86.3%) 3.4
4	ObjDom-ObjInv	$\mathbf{Dom}(R_o, X)$ $\wedge$ $\mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Rng}(S_o, X)$	41/51(80.4%) 3.7	39/51(76.5%) 3.2	44/51(86.3%) 3.6
5	ObjVal-ObjVal-DifInd-ObjFun	$X \sqsubseteq \exists R_o.\{i\}$ $\wedge$ $X \sqsubseteq \exists R_o.\{j\}$ $\wedge$ $\mathbf{Dif(i,j)}$ $\wedge$ $\mathbf{Fun(R_o)}$ $\rightarrow X \sqsubseteq \bot$	36/51(70.6%) 3.6	28/51(54.9%) 3.6	
6	ObjRng-ObjInv	$\mathbf{Rng}(R_o, X)$ $\wedge$ $\mathbf{Invs}(R_o, S_o)$ $\rightarrow \mathbf{Dom}(S_o, X)$	36/51(70.6%) 3.5	38/51(74.5%) 3.3	44/51(86.3%) 3.6
7	ObjMin-ObjFun	$X \sqsubseteq \geq nR_o.Y$, $n > 1$ $\wedge$ $\mathbf{Fun(R_o)}$ $\rightarrow X \sqsubseteq \bot$	37/51(72.5%) 3.5	33/51(64.7%) 3.5 38/51(74.5%) 3.7	

by using a Friedman's test revealed statistically reliable differences among the three cases (χ^2=6.72, p=0.04). Pairwise comparisons using a Wilcoxon Signed Ranks test showed the subjects performed significantly better on explanations of strategy 4 than those of strategy 3 (Z=2.36, p=0.02), but equally well between the original explanations and those of strategy 3 (Z=0.42, p=0.97), the original explanations and those of strategy 4 (Z=1.57, p=0.17). This suggested that among the three cases 4 was the most suitable strategy for these rules.

Similarly, the comparison of *difficulty ranking* by strategies across the three rules (by using a Friedman's test) revealed statistically reliable differences among the three cases (χ^2=11.35, p=0.003). Pairwise comparisons using a Wilcoxon Signed Ranks test showed that the subjects found the original explanations significantly easier than those of strategy 3 (Z=3.20, p=0.001), and those of strategy 4 were significantly easier than those of strategy 3 (Z=2.04, p=0.04<0.05). This meant that the subjects perceived that strategy 3, which paraphrased an $\mathbf{Invs}(R_o, S_o)$ axiom as "If something R_o something then the latter S_o the former; additionally, if something S_o something then the latter R_o the former", as the most difficult strategy among the three cases—probably because this paraphrase was longer than others.

For rules 2, 5, and 7 in Table 8.6, we tested whether the new paraphrases for difficult

axioms improved subjects' performance and/or reduced their difficulty ranking on the baseline of the original explanations. We found no reliable differences in *subjects' performance* between the original explanation and explanation 3 in rules 2 (p=1.00), and 5 (p=0.06) (by using McNemar's tests). We also found no reliable differences in *difficulty ranking* between the two explanations in rules 2 (Z=0.86, p=0.39), and 5 (Z=0.51, p=0.61) (by using Wilcoxon Signed Ranks tests). Similarly, there were no reliable differences in *subjects' performance* (Cochran's Q test, Q(2)=2.21, p=0.33) as well as *difficulty ranking* (Friedman's test,χ^2=3.11, p=0.21>0.05) among the three explanations in rule 7.

In conclusion, we chose strategy 4—i.e., contextualising **Invs**(R_o, S_o) axioms—to generate explanations for rules 3, 4, and 6. For rules 2, 5, and 7, the original explanations would be used.

8.4 Discussion and Conclusions

We have investigated the effectiveness of special strategies for explaining deduction rules that are shown to be difficult in Chapter 7, and found evidence that strategies can be beneficial. Specifically, for rule 2 ('ObjAll') in Table 8.1 which yields, for instance, "Everything that has no rating at all is a good movie" from "A good movie is anything that has only four stars as ratings", strategy 1A helps cancel the presupposition caused by the word 'only' (that a good movie always has at least a four-star rating) by explicating the premise as "A good movie is anything that has no ratings at all, or has only four stars as ratings".

For rule 8 ('ObjSom-Bot-1'), which yields, for instance, "There are no children" from "Every child likes a dragon" and "There are no dragons", strategy 1A also helps to cancel the presupposition caused by the word 'every' (that there exist at least a child) by explicating the premise as "Every child likes a dragon, or there are no children at all". This strategy can help rules 5 ('ObjVal-ObjVal-DifInd-ObjFun') and 7 ('ObjMin-ObjFun') in the same way.

Rules 3 ('ObjAll-ObjInv'), 4 ('ObjDom-ObjInv'), and 6 ('ObjRng-ObjInv') contain an **Invs** axiom (verbalised as, for instance, ""X owns Y" means the same as "Y is owned by X"") in their premises. For these rules, strategy 4 which contextualises **Invs** axiom as ""X owns Y" means the same as "Y is owned by X", from which its follows that "a person

Table 8.7: Explanations for difficult deduction rules

ID	Name	Deduction Rule	Explanation Template
1	ObjAll	$X \equiv \forall R_o.Y$ $\rightarrow \forall R_o.\bot \sqsubseteq$	An X is anything that R_o nothing at all, or R_o only Ys. $\rightarrow$Everything that R_o nothing at all is an X.
2	DatSom-DatRng	$X \sqsubseteq \exists R_d.D_{r0}$ $\wedge$ **Rng**(R_d, D_{r1}) D_{r0} & D_{r1} are disjoint $\rightarrow X \sqsubseteq \bot$	Every X R_d a D_{r0}, or there are no Xs at all. Any value that something R_d is a D_{r1}. D_{r0} values are not D_{r1} values. $\rightarrow$Nothing is an X.
3	ObjAll-ObjInv	$X \sqsubseteq \forall R_o.Y$ $\wedge$ **Invs**(R_o, S_o) $\rightarrow \exists S_o.X \sqsubseteq Y$	Every X R_o only Ys. "X R_o Y" means the same as "Y S_o X", from which it follows that "an X R_o a Y" means the same as "a Y S_o an X". $\rightarrow$Everything that S_o an X is a Y.
4	ObjDom-ObjInv	**Dom**(R_o, X) $\wedge$ **Invs**(R_o, S_o) $\rightarrow$ **Rng**(S_o, X)	Anything that R_o something is an X. "X R_o Y" means the same as "Y S_o X", from which it follows that "an X R_o a Y" means the same as "a Y S_o an X". $\rightarrow$Anything that something R_o is an X.
5	ObjVal-ObjVal-DifInd-ObjFun	$X \sqsubseteq \exists R_o.\{i\}$ $\wedge$ $X \sqsubseteq \exists R_o.\{j\}$ $\wedge$ **Dif**$(\mathbf{i}, \mathbf{j})$ $\wedge$ **Fun**$(\mathbf{R_o})$ $\rightarrow X \sqsubseteq \bot$	Every X R_o i, or there are no Xs at all. Every X R_o j, or there are no Xs at all. i and j are different individuals. Everything R_o at most one thing. $\rightarrow$Nothing is an X.
6	ObjRng-ObjInv	**Rng**(R_o, X) $\wedge$ **Invs**(R_o, S_o) $\rightarrow$ **Dom**(S_o, X)	Anything that something R_o is an X. "X R_o Y" means the same as "Y S_o X", from which it follows that "an X R_o a Y" means the same as "a Y S_o an X". $\rightarrow$Anything that S_o something is an X.
7	ObjMin-ObjFun	$X \sqsubseteq \geq nR_o.Y$, $n > 1$ $\wedge$ **Fun**$(\mathbf{R_o})$ $\rightarrow X \sqsubseteq \bot$	Every X R_d at least n D_rs, or there are no Xs at all. Everything R_d at most one value. $\rightarrow$Nothing is an X.
8	ObjSom-Bot-1	$X \sqsubseteq \exists R_o.Y$ $\wedge$ $Y \sqsubseteq \bot$ $\rightarrow X \sqsubseteq \bot$	Every X R_o a Y, or there are no Xs at all. Nothing is a Y. $\rightarrow$Nothing is an X.

owns a pet" means the same as "a pet is owned by a person"" is found to be effective. For other rules and cases, the original explanations are used.

Based on these above analyses, we have worked out new explanation templates for the 8 difficult rules, as presented in Tables 8.7. There exist alternative methods to paraphrase difficult axioms as well as elucidate these rules. Further work is needed to investigate such methods and compare with ours.

Chapter 9

Generation of Explanations

This chapter describes the generation of English explanations for entailments from OWL ontologies. The generation process includes three phrases, each of which is described separately in a section of this chapter. Section 9.1 describes algorithms for verbalising individual axioms in OWL. Section 9.2 briefly describes algorithms for verbalising our deduction rules—both normal and difficult ones. How these verbalisations and a proof tree are combined into a full explanation for an entailment is described in Section 9.3. Finally, the implementation of our explanation system and the current user interface of the system are briefly described in Section 9.4.

9.1 Generation of Axiom Verbalisations

9.1.1 Lexicons for Atomic Entities

To generate verbalisations for axioms in an OWL ontology, the lexical entries of all atomic entities in the ontology, including named classes, individuals, properties, data types, and literal values, need to be first computed. We use the algorithm implemented in the SWAT verbaliser [PT10, WTP11] to compute this information based on either entities' labels or identifier names. If an entity has at least one label in English, this label is used as the input string to compute the lexical entry. Otherwise, its identifier name is used. From

Table 9.1: Categories for lexicons

Atomic Entity	**Category**	**Examples**
Class	NOUN	person, animal
Individual	NAME	John, London
Property	VERB	owns, has as height
Data type	NOUN	string, integer
Literal	NAME	false, “6 feet”

the input string, non-alphanumeric characters are eliminated, and the string is split into words.

As in the SWAT verbaliser [PT10, WTP11], each lexical entry is classified into a *category*. There are three categories for atomic entities, namely *noun*, *verb*, and *proper noun* (or *name*). The classification of a lexical entry into a category is done automatically based on the *type* of the entity, as summarised in Table 9.1.

Since identifier names and labels of atomic entities are asserted by human ontology developers for their own purposes but not for linguistics purposes, the resulting lexical entries are unsuitable for verbalisation in some cases. They may contain abbreviations (e.g., the noun “CS course”), be grammatically incorrect (e.g., the noun “parental advisory suggested” and the verb “has age”), or both (e.g., the noun “r rated”). Additionally, it is quite common that an entity may have multiple English labels, and some labels are better than others. Therefore, the capability of selecting the best English label among alternatives as well as correcting problematic lexical entries would help to improve the quality of the resulting verbalisations. However, this is a tricky problem and requires serious investigation. Since this is not the focus of the work presented in this thesis, we followed the approach of the SWAT verbaliser which provides only minimal improvement for the lexical entries. Specifically, we converted verb lexical entries of the form “has + NOUN” into “has as + NOUN” (e.g., “has height” to “has as height”), and “NOUN + of” into “is a/an NOUN of” (e.g., “owner of” to “is an owner of”). For proper noun lexical entries, we automatically inserted the article “the” in front of names with three or more words (e.g., “Golden Gate Bridge” to “the Golden Gate Bridge”).

Table 9.2: Templates for OWL constructors; the category 'S' means a sentence, 'NP' means a noun phrase, 'VP' means a verb phrase, 'DET' means a determiner, 'CNOUN' means a compound noun, 'CONJ' is a conjunction, 'RPNOUN' means a relative pronoun, 'ADV' means an adverb; the mode 's' of verbs and verb phrases means singular, and 'p' means plural; A and B are named classes, C and D are complex class expressions

OWL Constructor	**Category**	**Grammar**
$A \sqsubseteq B$	S	NP[DET('every'), NOUN(A)], VP$_s$(B)
$A \sqsubseteq C$	S	NP[DET('every'), NOUN(A)], VP$_s$(C)
$C \sqsubseteq D$	S	NP[CNOUN('everything'), RPNOUN('that'), VP$_s$(C)], VP$_s$(D)
$\top \sqsubseteq A$	S	NP[CNOUN('everything')], VP$_s$(A)
$A \sqsubseteq \bot$	S	NP[CNOUN('nothing')], VP$_s$(A)
		where:
		- VP$_s$(A) $\rightarrow$ VERB$_s$('is'), NP[DET('a'), NOUN(A)]
		- VP$_p$(A) $\rightarrow$ VERB$_p$('are'), NP[DET(''), NOUN(A)]
		- Templates for VP$_s$(C) and VP$_p$(C) vary depending on the type of C.
$A \sqcap B$	VP$_s$	VERB$_s$('is'), CONJ('both'), NP[DET('a'), NOUN(A)], CONJ('and'), NP[DET('a'), NOUN(B)]
	VP$_p$	VERB$_p$('are'), CONJ('both'), NP[DET(''), NOUN(A)], CONJ('and'), NP[DET(''), NOUN(B)]
$A \sqcap C$	VP$_s$	VERB$_s$('is'), NP[DET('a'), NOUN(A)], RPNOUN('that'), VP$_s$(C)
	VP$_p$	VERB$_p$('are'), NP[DET(''), NOUN(A)], RPNOUN('that'), VP$_p$(C)
$C \sqcap D$	VP$_s$	VP$_s$(C), CONJ('and'), VP$_s$(D)
	VP$_p$	VP$_p$(C), CONJ('and'), VP$_p$(D)
$\forall R_p A$	VP$_s$	VERB$_s$(R_p), ADV('only'), NP[DET(''), NOUN(A)]
	VP$_p$	VERB$_p$(R_p), ADV('only'), NP[DET(''), NOUN(A)]
$\forall R_p C$	VP$_s$	VERB$_s$(R_p), ADV('only'), CNOUN('things'), RPNOUN('that'), VP$_p$(C)
	VP$_p$	VERB$_p$(R_p), ADV('only'), CNOUN('things'), RPNOUN('that'), VP$_p$(C)

9.1.2 Templates for OWL Constructors

To generate verbalisations for OWL axioms, each OWL constructor is mapped into a *template* which corresponds to a component in English grammar. As in the SWAT verbaliser [PT10, WTP11], each class constructor is mapped to a verb phrase (VP) template, and each axiom constructor is mapped to a sentence (S) template. However, instead of mapping class constructors into singular-mode NPs and VPs (as in the SWAT verbaliser), we map them into phrases in both singular and plural modes in order to grammatically generate appropriate sentences and phrases in an explanation. As an example, Table 9.2 shows the mapping of three OWL constructors, one for axiom and two for class constructors, to templates for English. In this way, the verbalisation for an OWL axiom, or even an OWL class expression, can be generated in the form of a syntactic tree. This is beneficial for explanation purposes as verbalisations can be modified easily to fit into the final explanations.

9.2 Generation of Explanations for Steps

Generally speaking, there are two approaches to scientific explanations, namely *top-down* and *bottom-up* [Fri74, Sal84, Kit85]. According to Kitcher [Kit85], the top-down approach proceeds toward "the identification of causes" in particular states or events whereas the bottom-up approach proceeds toward "stitching together results about the causation of individual states and events". This classification is also true in our case. Given a deduction, two types of explanations can be generated: a top-down explanation starts from the conclusion to the premises of the rule, whereas a bottom-up explanation starts from the premises to the conclusion. Which is best might depend on circumstances. However, as the first attempt to this problem, we focus only on generating a top-down explanation for a rule.

Lipton [Lip01] argued that one feature of good scientific explanations is the distinction between "knowing that a phenomena occurs and understanding why it does"—which is similar to the distinction between grasping the conclusion and understanding the explanation of why it follows from the premises. Therefore, we include both of these separately in an explanation. The explanation template for each deduction is based on the verbalisation of the rule worked out in Chapter 6. For the ten most difficult rules identified in Chapter 8, their explanation templates follow the verbalisations worked out in Chapter 8. As an example, the following template is what the top-down explanation for rule 'ObjSom-ObjDom' (rule 36 in Table 4.3) looks like:

> The statement "every X is a Y" is implied because:
> - every X R_o a Z, and
> - anything that R_o something is a Y.

As will be explained shortly, nodes within a proof tree, and so those of individual inference steps, are labelled with "axiom 1", "axiom 2" etc. (for terminal nodes or lemmas) and "(a)", "(b)", etc. (for non-terminal nodes or premises), before generating an explanation. In an inference step associated with rule "ObjSom-ObjDom", for example, if the conclusion and the second premise are labeled with "(a)" and "(b)", and the first premise is labelled with "axiom 1", then its explanation template would be (the premise labelled with "(b)"

is a non-terminal node and its explanation will be presented after this explanation):

> Statement (a) is implied because:
> - every X R_o a Z (from axiom 1), and
> - anything that R_o something is a Y (b).

If both of the premises are terminal nodes and are labelled with "axiom 1" and "axiom 2" (both are premises), then the explanation template for this inference step would be:

> Statement (a) is implied because:
> - every X R_o a Z (from axiom 1), and
> - anything that R_o something is a Y (from axiom 2).

9.3 Generation of Complete Explanations

In our work, an explanation is generated from a given proof tree. As in the generation of explanations for deduction rules, there are two approaches for the generation of explanations for proof trees, namely top-down and bottom-up. In the scope of this thesis, we focus only on *top-down* explanations in which explanations for individual steps also conform to the top-down approach.

The main algorithm **GenerateTopDownExplanation(P)** for generating an explanation is summarised in Algorithm 3, where P is the input proof tree. It first traverses the tree to compute appropriate labels, one for each node in the tree. The purpose of these labels is to help readers locate the statements quicker as well as to link the statements with the premises in the original justification. Specifically, labels for terminal nodes are in the form of "axiom n" where n is the index of the associated axiom in the original justification, and labels for non-terminal nodes are of the form "(a)", "(b)", and so on.

Algorithm 3 GenerateTopDownExplanation(P)

1: $Explanation_{complete} \leftarrow \{\}$
2: ComputeTopDownLabelsByNode($P.root$)
3: ComputeATopDownExplanationByNode($P.root$, $Explanation_{complete}$)
4: **return** $Explanation_{complete}$

Based on the computed labels, which are stored within the nodes, our main algorithm then traverses the tree again to generate explanations for individual inference steps, one at each non-terminal node, based on the associated deduction rules. With this design, explanations for individual steps can be flexibly modified in order to provide a good explanation. Our algorithm to generate individual explanations is summarised in Algorithm 4.

Algorithm 4 ComputeTopDownExplanationByNode(*Node*,*Explanation*)

```
1: if Node! = null and !Node.isTerminal() then
2:     Rule ← Node.getAppliedRule()
3:     if Rule! = null then
4:         SortedChildNodes ← Rule.sortPremises(Node.getChildren())
5:         Explanation_incomplete ← Rule.computeATopDownExplanation(Node, SortedChildNodes)
6:         Explanation.add(Explanation_incomplete)
7:         for ChildNode ∈ SortedChildNodes do
8:             ComputeTopDownExplanationByNode(ChildNode, Explanation)
9:         end for
10:    end if
11: end if
```

9.4 Implementation and User Interface

We have implemented a Java program that can automatically generate one or more explanations for a given entailment of an OWL ontology. This program is currently a plug-in of the SWAT verbaliser [PT10, WTP11], so ontology developers can use it as a tool for checking and debugging while developing their ontologies. The number of explanations for an entailment depends on the number of its justifications—for each justification, only one explanation (that the program identifies as the most understandable among a set of alternatives) is shown to end-users.

The main steps of the generation of explanations are described in Figure 1.2. The program starts from the justifications of the entailment, which can be computed using the module implemented by Horridge [KPHS07]. Thereafter, it constructs one or more proof trees for each justification by using the rule set described in Chapter 4. This program retains all of the justifications of the entailment because each of them may show a different reason why the entailment follows from the ontology. [1] Recall that our deduction rules may result in zero, one, or multiple proof trees for a given justification, but only one tree is selected

[1] One might argue that it is better to select only one best proof tree, and so only one best explanation for each entailment. However, we find that in many cases, different justifications of a problematic entailment show different errors or mistakes. This information might suggest different corrections for the ontology, so is useful for the repairing purposes. At this stage, it is up to the readers to decide which justifications and/or explanations are useful for them.

for each justification—if one tree is found then it is selected; if multiple trees are found, the program uses the understandability model (described in Chapter 7) to select the most understandable one. In the case of no tree being found (because of the limited coverage of the rule set), the system simply presents the justification in which both the conclusion and the premises are verbalised. From each selected proof tree, the system plans and generates an English explanation.

As discussed in Section 5.3, the exhaustive search algorithm used in the algorithm for constructing proof trees is a computational limitation of our system. Therefore, we set a time out of 60 seconds for this computation in order to guarantee the practical use of our explanation system as a plug-in for interactive real-time debugging purposes. Consequently, there would be cases in which no proof tree would be computed because of this time out setup. In such cases, the system would simply present the justification in which both the conclusion and the premises are verbalised.

In the output interface, each justification is presented in a tab. These justifications are first sorted by whether they have proof trees—that is, justifications that have at least a proof tree will precede those that have no proof trees. Among the former group, justifications are sorted in order of decreasing understandability (of their best proof trees)—i.e., the easiest one is shown first. Currently, both the proof trees and the English explanations are presented in the interface. Figure 9.1 shows a screen shot of the output interface of our program for an entailment that has three justifications.

9.5 Conclusions and Future Work

We have designed and implemented a program that can automatically generate one or more English explanations for an entailment from an OWL ontology, one based on each justification. Currently, an explanation is generated in the form of a paragraph that explains *all* inference steps linking from the premises to the conclusion but in the top-down approach—i.e., stating the conclusion then explaining why it follows. For large justifications, explanations generated in this way are still long, and may explain a repeated rule multiple times (when the rule occurs twice or more times within a proof tree). Therefore, in addition to enhancement of proof trees, enhancement of the planning are required to make the final explanations concise, fluent, and readable, and the main inference steps

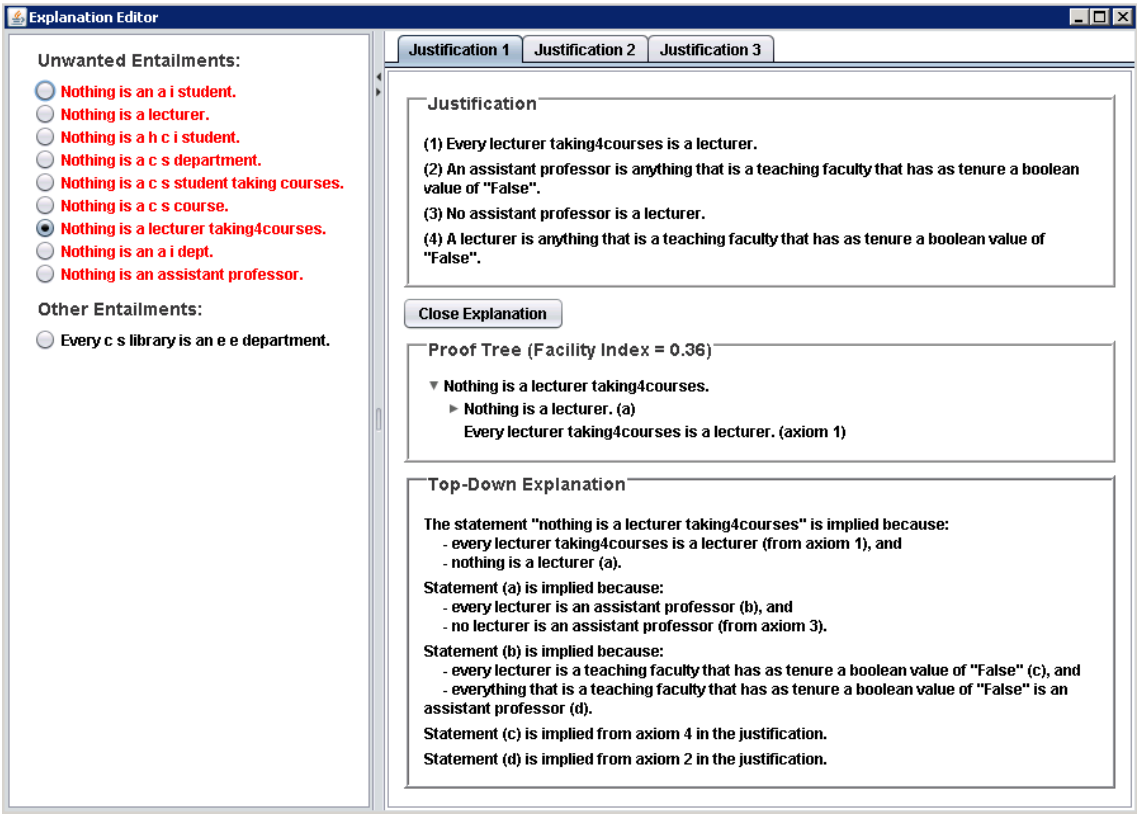


Figure 9.1: A screen shot of the output of our explanation system for the entailment $LectureTaking4Course \sqsubseteq \bot$ of the university ontology

more visible for end-users.

Chapter 10

Preliminary Evaluation

For the evaluation of our explanation tool, we want to find out (a) whether it is helpful for domain experts as well as knowledge engineers in debugging OWL ontologies (compared to the baseline of justifications for entailments), and (b) if so, how the final explanation is best set up (i.e., whether the proof tree or the English explanation is best, or we need to give both of them).

As in the study conducted by Kalyanpur et al. [KPSH05, Kal06] to evaluate debugging features implemented in the Swoop ontology editor, designing a serious study towards these purposes is very difficult. Before the two above-mentioned questions can be investigated, at least all of the following conditions must be met:

Thorough training It is crucial that the users, whether domain experts or knowledge engineers, are familiar with reasoning with OWL ontologies as well as how to use the explanation tool for debugging purposes. This means that they need to be thoroughly trained on these skills before participating in the study.

High standard of verbalisation It is also crucial that the users can understand exactly what each axiom or entailment means. This holds whether the axiom or entailment is expressed in English. A serious study would therefore require test ontologies to have a very high standard of English verbalisation[1].

[1] This condition was not considered in Kalyanpur et al.'s study [KPSH05, Kal06].

Familiar topics It is also crucial that the users can distinguish certainly correct axioms from uncertain ones. Without this condition, people have no way to identify which axioms may contain mistakes, as well as how to correct them. This condition requires the users to have a relatively good understanding of the domain. Therefore, test ontologies should be in topics that are familiar to the users.

Additionally, justifications need to vary in both size and complexity in order to identify in which cases explanations are helpful for the users. The think-aloud protocol, or recording (e.g., video recording or eye-tracking recoding) should be employed to determine whether users understand and utilize explanations. Since this study focuses on the debugging process, our test ontologies should contain mistakes that lead to one or more undesired entailments, some of which are used as test entailments.

Given the complexity of designing and conducting a serious study, in the scope of this thesis, we conduct just a preliminary evaluation study to collect users' comments on different types of explanations as well as the design of the study (e.g., the selection of test ontologies, entailments, verbalisations etc.). Specifically, we want to compare subjects' performance as well as their difficulty ranking on ontology debugging tasks when one of the following five types of explanations for entailments is provided:

1. Only the entailment and its justification (verbalised in English)
2. A proof tree in which all nodes are verbalised but difficult steps are not further elucidated
3. An English explanation in which difficult steps are *not* further elucidated
4. A proof tree with all nodes verbalised and difficult steps highlighted and elucidated
5. An English explanation in which difficult steps are further elucidated—i.e., the explanation generated by our system

Among these explanation types, the first one is our baseline; the remaining four are to compare between proof trees or English explanations in two cases: with and without further elucidations for difficult steps. This chapter describes the set-up and results of this study, followed by the discussion of what have been learnt from it for serious evaluation studies that will be conducted in the future.

10.1 Materials

As mentioned before, the aim of this study is to compare subjects' performance and their difficulty rankings, and collect their comments on ontology debugging tasks when five different types of explanations are provided. As test ontologies, we selected five real world ontologies of various but familiar topics that included undesired entailments (i.e., incorrect entailments) from our corpus (described in Chapter 4). They are:

1. "Movie" ontology, which describes categories of movies
2. "University" ontology, which describes positions in universities
3. "Multi-modal Yard" ontology, which describes the physical and abstract elements of a yard such as containers, devices etc.
4. "Particle" ontology, which describes basic particles such as neutrons, protons etc.
5. "Koala" ontology, which describes features of koalas

From each ontology, an undesired entailment and one of its justification were selected as a test inference. These test inferences are listed in Table 10.1.

For each test ontology and the associated test inference, we devised *five* test problems. In each problem, part of the ontology (that caused the undesired entailment to follow)[2], the entailment, and an explanation were given in English, and the subjects were asked whether the explanation makes it clear why the entailment was literally drawn from the ontology by a computer program. The subjects were also asked to rank how difficult they found the question on a scale from 5 (very easy) to 1 (very difficult), identify and correct the axiom(s) that they thought to be the main cause of the entailment, and finally, provide their comments on the overall problem. The only difference between the five problems (from the same test ontology and inference) lay in the explanation—i.e., in each problem, the explanation was presented in one of the five types mentioned above. Since there were five test inferences, a total of 25 test problems were devised. As an example, Figures 10.1-10.5 show five explanations of the five types for the undesired entailment "Every rating is a movie" of the Movie ontology.

[2]We stipulated that exactly ten axioms from an ontology would be given.

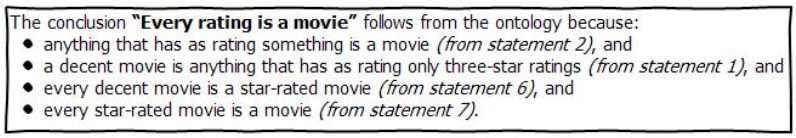


Figure 10.1: The baseline explanation for the undesired entailment "Every rating is a movie" of the Movie ontology, given in the form of the verbalisation of the entailment and its justification

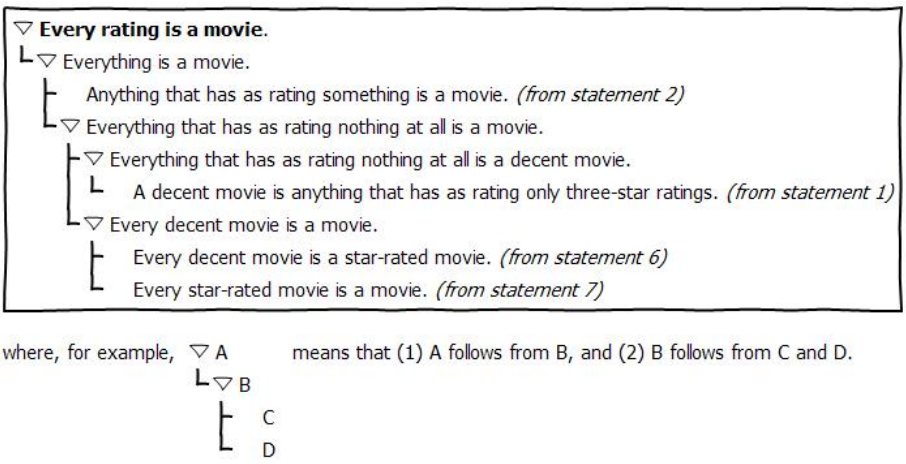


Figure 10.2: The explanation for the undesired entailment "Every rating is a movie" of the Movie ontology, given in the form of a proof tree, but without further elucidations for difficult inferences

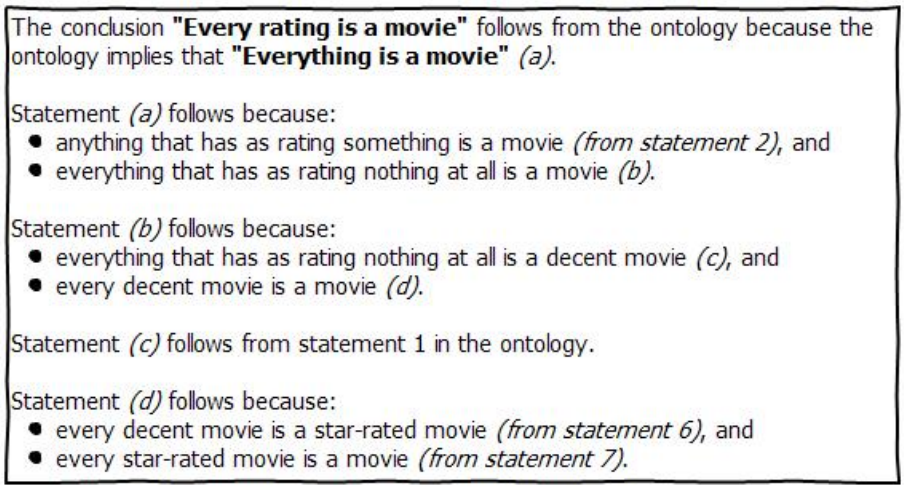


Figure 10.3: The explanation for the undesired entailment "Every rating is a movie" of the Movie ontology, given in the form of an English explanation, but without further elucidations for difficult inferences

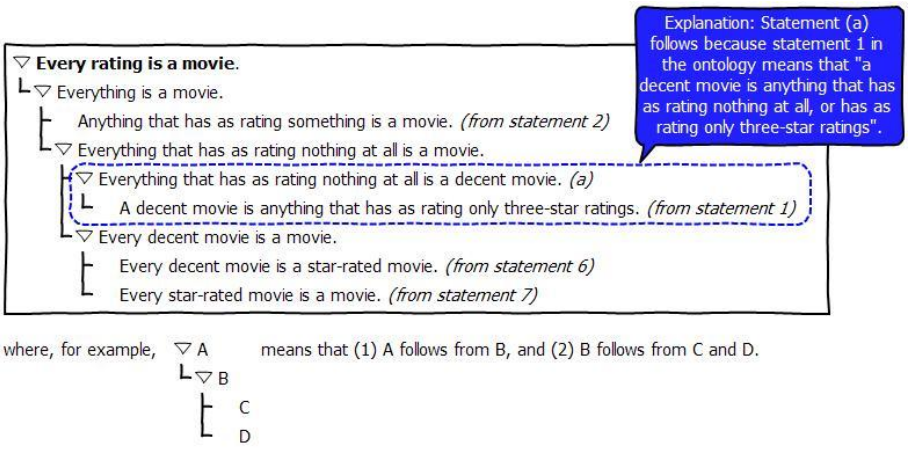


Figure 10.4: The explanation for the undesired entailment "Every rating is a movie" of the Movie ontology, given in the form of a proof tree with further elucidations for difficult inferences, where possible

The conclusion **"Every rating is a movie"** follows from the ontology because the ontology implies that **"Everything is a movie"** *(a)*.

Statement *(a)* follows because:
- anything that has as rating something is a movie *(from statement 2)*, and
- everything that has as rating nothing at all is a movie (*b*).

Statement *(b)* follows because:
- everything that has as rating nothing at all is a decent movie *(c)*, and
- every decent movie is a movie *(d)*.

Statement *(c)* follows because statement 1 in the ontology means that "a decent movie is anything that has as rating nothing at all, or has as rating only three-star ratings".

Statement *(d)* follows because:
- every decent movie is a star-rated movie *(from statement 6)*, and
- every star-rated movie is a movie *(from statement 7)*.

Figure 10.5: The explanation for the undesired entailment "Every rating is a movie" of the Movie ontology, given in the form of an English explanation with further elucidations for difficult inferences, where possible

Table 10.1: The list of inferences tested in the preliminary evaluation study, the indexes used in the justifications are the indexes of the axioms in the test ontologies used in the study

ID	Ontology	Entailment	Justification
1	Movie	$Rating$ $\sqsubseteq$ $Movie$	1.$DecentMovie \equiv \forall hasRating.ThreeStar$ 2.**Dom**$(hasRating, Movie)$ 6.$DecentMovie \sqsubseteq StarRatedMovie$ 7.$StarRatedMovie \sqsubseteq Movie$
2	University	$HCIStudent$ $\sqsubseteq$ $\bot$	5.$HCIStudent \sqsubseteq \exists researchesInto.HCITopic \sqcap \exists supervisedBy.CSProfessor$ 6.$CSProfessor \sqsubseteq FacultyMember \sqcap \forall supervises.AIStudent$ 9.**Invs**$(supervises, supervisedBy)$ 10.**Dis**$(HCIStudent, AIStudent)$
3	Multi-modal Yard	$OfficeBuilding$ $\sqsubseteq$ $\bot$	1.**Invs**$(mountedOn, hasDevice)$ 2.**Dom**$(mountedOn, SupplementaryDevice)$ 3.$FireAlarmSystem \sqsubseteq \exists hasDevice.SmokeAlarm$ 9.$OfficeBuilding \sqsubseteq \exists equippedWith.FireAlarmSystem$ 10.**Dis**$(SmokeAlarm, SupplementaryDevice)$
4	Particle	$TetraNeutronNucleus$ $\sqsubseteq$ $\bot$	2.$Neutron \sqsubseteq= 1 hasPart.UpQuark$ 5.$TetraNeutronNucleus \sqsubseteq= 4 madeUpOf.Neutron$ 9.$UpQuark \sqsubseteq= 2 hasElectricalCharge.PositiveCharge$ 10.**Fun**$(hasElectricalCharge)$
5	Koala	$NorthernKoala$ $\sqsubseteq$ $\bot$	1.$Koala \sqsubseteq Marsupial$ 2.$Koala \sqsubseteq \exists hasHardWorkingAttribute.\{False\}$ 5.$NorthernKoala \equiv Koala \sqcap lives.NorthernAreaInAustralia$ 7.**Dom**$(hasHardWorkingAttribute, Person)$ 10.**Dis**$(Marsupial, Person)$

Table 10.2: Distribution of test problems into five studies, the number in each cell is the identifier of the associated study

	Explanation 1	**Explanation 2**	**Explanation 3**	**Explanation 4**	**Explanation 5**
Inference 1	1	2	3	4	5
Inference 2	5	1	2	3	4
Inference 3	4	5	1	2	3
Inference 4	3	4	5	1	2
Inference 5	2	3	4	5	1

As in previous studies, *control questions* are used to filter 'scammers'. In this study, a control question is an extra question presented within a test problem, after the ontology but before the test questions. In this question, the subjects are asked whether a statement can be literally inferred from the ontology. The inference in a control question is often trivial, and the correct answer can be either "Yes" or "No". Figure 10.6 shows a test problem used in our study. In this problem, the first question is the control one, and its correct answer is "No"; the explanation is of the first type—that is, only the entailment and its justification (verbalised in English) are provided.

Since there were 25 test problems and each problem was quite long, we divided them into five smaller studies, each of which included five test problems. The distribution of the test problems into studies is shown in Table 10.2. In this way, none of the subjects could work on more than one test problem of either the same explanation type or the same test inference, so the ordering side effect could be avoided. Additionally, subjects' performance and their difficulty ranking by explanation types or test inferences were comparable.

In addition to the test problems, three *control problems*, as opposed to *test problems*, were devised in order to enable the use of CrowdFlower's quality control service. These control problems were designed to resemble test problems but all questions (both control and test) were obvious to subjects who did the test seriously. These problems were devised from three additional ontologies of familiar topics, namely "Pizza", "Economy", and "Car" (created by ourselves). As an example, Figure 10.7 shows a control problem used in this evaluation.

In summary, we conducted five small studies, each of which consisted of five test problems and three control problems. Among the eight control questions in each study (in both test and control problems), three of them had the correct answer "Yes", and the remaining questions had the correct answer "No".

PROBLEM 3:

An "ontology" in computer science is a set of logic-based statements from which a computer program can draw conclusions. Here is an ontology of ten statements, translated into English:

1. **A decent movie is anything that has as rating only three-star ratings.**
2. **Anything that has as rating something is a movie.**
3. **A good movie is anything that has as rating only four-star ratings.**
4. **Every three-star rating is a rating.**
5. **Every four-star rating is a rating.**
6. **Every decent movie is a star-rated movie.**
7. **Every star-rated movie is a movie.**
8. **Every director is a person.**
9. **James Cameron is a director.**
10. **Anne Hathaway is an actor.**

When a computer program draws conclusions from an ontology, it interprets the statements in a strictly literal way.

From the ten statements above, can the computer program prove that "Anne Hathaway is a person"?

- ○ **Yes**
- ○ **No**

Surprisingly, the computer program **correctly** draws the conclusion "Every rating is a movie" from the ontology. Why? Consider the following explanation:

> The conclusion **"Every rating is a movie"** follows from the ontology because:
> - anything that has as rating something is a movie *(from statement 2)*, and
> - a decent movie is anything that has as rating only three-star ratings *(from statement 1)*, and
> - every decent movie is a star-rated movie *(from statement 6)*, and
> - every star-rated movie is a movie *(from statement 7)*.

- Does this explanation make it clear why the computer program concluded "Every rating is a movie" from the ontology?

- ○ **Yes**
- ○ **No**

- How difficult did you find to understand this explanation?

- ○ **Very easy**
- ○ **Easy**
- ○ **Average**
- ○ **Difficult**
- ○ **Very difficult**

- Which statement from the ontology should be corrected to avoid the conclusion "Every rating is a movie"? Choose one from the following options:

- ○ **None of the statements**
- ○ **Statement 1: A decent movie is anything that has as rating only three-star ratings.**
- ○ **Statement 2: Anything that has as rating something is a movie.**
- ○ **Statement 3: A good movie is anything that has as rating only four-star ratings.**
- ○ **Statement 4: Every three-star rating is a rating.**
- ○ **Statement 5: Every four-star rating is a rating.**
- ○ **Statement 6: Every decent movie is a star-rated movie.**
- ○ **Statement 7: Every star-rated movie is a movie.**
- ○ **Statement 8: Every director is a person.**
- ○ **Statement 9: James Cameron is a director.**
- ○ **Statement 10: Anne Hathaway is an actor.**

- If you found that a statement in the ontology should be corrected to avoid the conclusion "Every rating is a movie", please briefly describe how you would correct it. Otherwise, please enter "None" in the following textbox.

- If you found the above questions difficult or confused, could you explain why? (optional)

Figure 10.6: A test problem devised for the undesired entailment "Every rating is a movie" from the Movie ontology

PROBLEM 6:

An "ontology" in computer science is a set of logic-based statements from which a computer program can draw conclusions. Here is an ontology of ten statements, translated into English:

1. **"X is a topping of Y" means the same as "Y has as topping X".**
2. **A cheese pizza is anything that is a pizza that has as topping a cheese topping.**
3. **An interesting pizza is anything that is a pizza that has as topping at least four toppings.**
4. **Anything that has as topping something is a pizza.**
5. **Every American pizza is a pizza.**
6. **Every American pizza has as topping a mozzarella topping.**
7. **Every ice cream has as topping a fruit topping.**
8. **No cheese topping is a fruit topping.**
9. **No ice cream is a pizza.**
10. **Every mozzarella topping is a cheese topping.**

When a computer program draws conclusions from an ontology, it interprets the statements in a strictly literal way.

From the ten statements above, can the computer program prove that "Every American pizza is an interesting pizza"?

- ○ **Yes**
- ○ **No**

Surprisingly, the computer program **correctly** draws the conclusion "Nothing is an ice cream" from the ontology. Why? Consider the following explanation:

> The conclusion **"Nothing is an ice cream"** follows from the ontology because:
> - every ice cream has as topping a fruit topping *(from statement 7)*, and
> - anything that has as topping something is a pizza *(from statement 4)*, and
> - no ice cream is a pizza *(from statement 9)*.

- Does this explanation make it clear why the computer program concluded "Nothing is an ice cream" from the ontology?

- ○ **Yes**
- ○ **No**

- How difficult did you find to understand this explanation?

- ○ **Very easy**
- ○ **Easy**
- ○ **Average**
- ○ **Difficult**
- ○ **Very difficult**

- Which statement from the ontology should be corrected to avoid the conclusion "Nothing is an ice cream"? Choose one from the following options:

- ○ **None of the statements**
- ○ **Statement 1: "X is a topping of Y" means the same as "Y has as topping X".**
- ○ **Statement 2: A cheese pizza is anything that is a pizza that has as topping a cheese topping.**
- ○ **Statement 3: An interesting pizza is anything that is a pizza that has as topping at least four toppings.**
- ○ **Statement 4: Anything that has as topping something is a pizza.**
- ○ **Statement 5: Every American pizza is a pizza.**
- ○ **Statement 6: Every American pizza has as topping a mozzarella topping.**
- ○ **Statement 7: Every ice cream has as topping a fruit topping.**
- ○ **Statement 8: No cheese topping is a fruit topping.**
- ○ **Statement 9: No ice cream is a pizza.**
- ○ **Statement 10: Every mozzarella topping is a cheese topping.**

- If you found that a statement in the ontology should be corrected to avoid the conclusion "Nothing is an ice cream", please briefly describe how you would correct it. Otherwise, please enter "None" in the following textbox.

- If you found the above questions difficult or confused, could you explain why? (optional)

Figure 10.7: A control problem devised for the undesired entailment "Nothing is an ice cream" from the ontology "Pizza.owl"

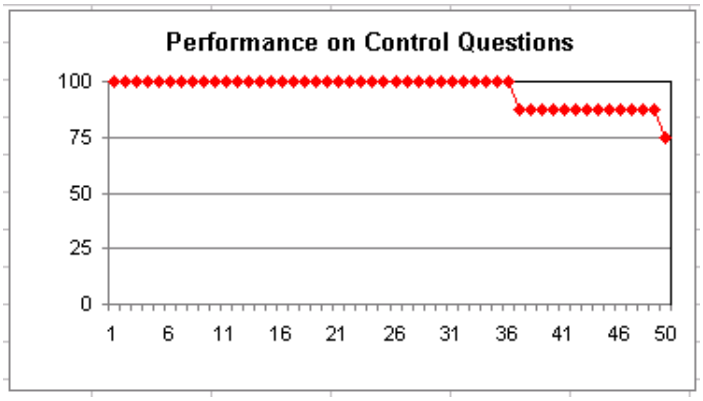


Figure 10.8: Subjects' performance on control questions in all five studies

10.2 Method

We used the usual procedure that was applied to all our previous studies. In our studies, each subject was required to answer all questions in eight problems of the associated study, and no subject was allowed to participate in more than one study. We set up to collect the responses of exactly 10 subjects for each study, and so 50 subjects in total. Most of them reported as having a basic understanding of logic, but not being familiar with OWL.

10.3 Control Questions

Figures 10.8 shows that for the 50 subjects that participated in our five studies, all of them answered 75% or more of the control questions correctly, suggesting that they were performing the tasks seriously.

10.4 Results

The task required subjects to make three judgements, which we analysed separately: (1) whether the explanation was useful; (2) which axiom was problematic; and (3) how this axiom should be corrected. It should be noted that for each test inference the main inference that caused the undesired entailment to follow always involved two axioms in the ontology. Therefore, when analysing the identification of the problematic axiom(s), we accepted all answers that selected any of these axioms, or both of them. Similarly, when analysing the corrections, changes of any kind that could help to cancel the undesired entailment were accepted, regardless of whether the resulting statements were true. However, we classi-

fied subjects' corrections into four categories—they are: (i) change the axiom correctly, (ii) remove the axiom correctly, (iii) provide no correction, and (iv) make inappropriate correction.

The results are summarised in Table 10.3. Overall, we found *no* evidence of statistically reliable differences among the five conditions, perhaps because of the complications mentioned in the introduction to the chapter and the small number of participants in each study. However, we found *improvement* when comparing explanation type 5 (i.e., text explanations with elucidation—they were outputs of our explanation system) and type 1 (i.e., justifications verbalised—they were the baseline). In particular, there were more participants who found text explanations with elucidation useful for understanding undesired entailments (41 participants for type 5 versus 35 participants for type 1); there were more participants who fixed the mistaken axioms correctly (39 participants for type 5 versus 33 participants for type 1). Moreover, even though text explanation with elucidation were often longer than verbalised justifications, most participants found that their difficulty levels were similar (mean difficulty value of 2.7 for type 5 versus 3.0 for type 1). This suggested a positive result on whether English explanations generated by our system would be useful for debugging OWL ontologies.

Table 10.3: Results of the preliminary evaluation study, "#Useful" means the number of subjects finding the given explanation useful, "Mean Difficulty" means the mean of ranked difficulties (from 5 (very easy) to 1 (very difficult)), "#Change Correctly" means the number of subjects changing the axiom correctly, "#Remove Correctly" means the number of subjects removing the axiom correctly, "#No Correction" means the number of subjects providing no correction, and finally "#Change Inappropriately" means the number of subjects changing the axiom incorrectly

Explanation Type	**#Useful**	**Mean Difficulty**	**#Change Correctly**	**#Remove Correctly**	**#No Correction**	**#Change Inappropriately**
1 (Justification)	35	3.0	27	6	5	0
2 (Proof Tree)	41	3.0	35	6	3	4
3 (Text)	39	2.7	24	9	5	5
4 (Proof Tree & Elucidation)	33	2.6	26	10	2	2
5 (Text & Elucidation)	41	2.7	32	7	7	1

10.5 Subjects' Comments and Lessons Learnt

Throughout the study, we also obtained a number of subjects' comments on both the design of the studies (e.g., the selection of test ontologies, entailments, verbalisations etc.), as well as how the final explanation should be best set up. These comments are listed in Appendix B. Generally speaking, the subjects' comments fall into the following categories:

1. Comments about specific *axioms*—e.g., that an axiom is unclear (such as comments 3 in 1.1, 2 in 4.1, 1 and 2 in 5.2, etc. in Appendix B)

2. Comments about *ontology topics*—e.g., that lack of domain knowledge make it difficult to follow the reasoning (such as comments 1 in 4.2 and 1 in 4.3), but familiarity causes the use of subjects' own assumption (such as comments 1 in 4.3)

3. Comments about specific *inference steps*—e.g., that an inference step is false (such as comments 3 in 1.2, 5 in 1.3 etc.), or too difficult (such as comments 1, 3, and 4 in 1.4)

4. Comments about *explanation presentations*—e.g., that English explanations are difficult to understand (such as comments 3 in 2.3 and 3 in 5.3), that diagrams are better than text (such as comments 4 in 4.2 and 3 in 5.3), or vice-versa (such as comment 2 in 3.4)

For category (1), it seems clear that some axioms are simply not understood, mostly because of weird English wordings. Phrases such as "has as rating only three-star ratings" and "has as hard-working attribute some value" do no appear to be grammatically correct, and so are misunderstood by most people[3]. If a premise or the conclusion of an inference is not well-understood, the inference might appear invalid even if the correct logic is applied. Perhaps ontologies that are developed using a natural language editor, or those of a domain that has a grammar specially developed for it might help.

For category (2), the technical subject-matter of some test ontologies is still difficult for untrained people to conceptualize concepts and reason with them. Terms likes "tetra-neutron nucleus" and "up-quark" in the "Particle" ontology, for example, are so technical

[3]Verbalising OWL statements containing property names such as "hasRating" and "hasHardWorkingAttribute" is a research topic in ontology verbalisation.

that they might as well be nonsense words—they should be avoided in the future study. Providing training to the participants is necessary to help them understand the context of the test.

For category (3), it seems clear that the inference based on the trivial satisfaction of the universal restriction in OWL is so difficult that some participants judge that it was drawn by the computer program by mistake. As before, training might help to avoid such misunderstanding.

For category (4), comments vary, suggesting that different people like different presentations of explanations, and the important thing is to cater for a variety of preferences. Perhaps presenting the final explanations in both forms, as our explanation tool currently does, is the best solution for this purpose.

In addition to the above-mentioned analyses, the subjects' corrections and comments also suggest that there exist cases in which mistakes in axioms are so obvious that the subjects tend to fix these axioms locally without reading and utilizing the explanations. To detect such cases, robust data gathering methods such as think aloud protocol and recording would be helpful.

10.6 Related Work

Kalyanpur et al. [KPSH05, Kal06] conducted a study to evaluate debugging features implemented in the Swoop/Pellet ontology editor. The subjects of this study were twelve subjects that had experience with OWL and a good understanding of description logic reasoning. Before the test, the subjects were trained on OWL, Swoop, and the debugging tools. In the main test, the subjects were divided into four groups, each of which performed a number of debugging tasks in a different condition—specifically, one group with no debugging support, and each of the remaining groups with a different type of debugging support. A weakness of this study was that it did not control for whether the subjects were familiar with the concepts and knowledge within the test ontologies.

10.7 Conclusions and Future Work

We have conducted a preliminary study to find out (a) whether our explanation tool is helpful for debugging tasks in OWL ontologies (compared to the baseline of justifications for entailments), and (b) how the final explanation is best set up (i.e., proof trees versus English text). Although no reliable evidence of differences are found due to the limited scope of this study, we have learnt a number of valuable lessons from subjects' comments (listed in Appendix B) in selecting the appropriate presentation for our explanations—that is, presenting both proof trees and English explanations to cater for a variety of preferences—as well as designing a serious evaluation study in the future—e.g., selecting only familiar ontologies with a very high standard of verbalisation, providing training to participants, using robust data gathering methods etc. This means that although we have been able to test our ideas in theory, using the studies on the understandability of single and two-step inferences, the conditions are not yet right to test them in practice—at least not in the context of a serious ontology development project.

Chapter 11

Conclusion

This thesis advances knowledge and techniques in generating explanations for entailments in OWL ontologies. In the practical limitation of this thesis, we restrict to explanations for non-trivial subsumption entailments between two class names—i.e., entailments of the forms $\top \sqsubseteq A$ (Everything is an A), $A \sqsubseteq \bot$ (Nothing is an A), and $A \sqsubseteq B$ (Every A is a B), where A and B are class names. We use a set of deduction rules collected from a corpus-based study of subsumption inferences in OWL ontologies to guide the generation of explanations, focus on the indentification of difficult inferences and how to explain them to end-users (instead of maximising the coverage of the rule set). Empirical studies are used to assess the cognitive difficulty of OWL inferences, but we test with only ordinary people who have limited knowledge of OWL and logic.

To conclude, this chapter first summarises our answers to the research questions set out in the Introduction (in Section 11.1), and other key findings (in Section 11.2). The limitations of the research are discussed in Section 11.3, followed by the discussion of the broader significance made by this thesis in Session 11.4. Finally, questions raised for further research in the future are discussed in Section 11.5.

11.1 Answers to Research Questions

Our answers for the research questions set out in the Introduction are as follows:

Question 1: How can we find deduction patterns that are suitable for single inference steps in an explanation (considering both frequency and level of understandability)?

Answer: We have found suitable patterns by identifying the most common deduction patterns in a large corpus of real world ontologies, and choosing those that are frequent and cannot be usefully simplified further.

Question 2: Given a set of alternative multi-step explanations, how can we provide an empirically grounded criterion for deciding which is most suitable—i.e., which would be understood best by our target users, assumed to be non-logicians?

Answer: We have provided an empirically grounded criterion for choosing the most understandable proof tree (among a set of alternatives) by studies measuring the understandability of our single-step inferences, and validating a method for combining these measures for multi-step inferences.

Question 3: Having determined the logical structure of an explanation, how can we most clearly express it in English through appropriate ordering, verbalisation, and extra elucidation?

Answer: We have used existing work on ontology verbalisation and the psychology of reasoning to inform decisions on wording and organisation, along with an extra study on specific issues (e.g., how to express disjoint and equivalent classes). We have also used the understandability measures (see the second answer) in order to decide which inference steps require further elucidation, and test empirically which of various possible strategy-based elucidations work best.

11.2 Other Key Findings

In addition to the above-mentioned answers, our key findings are:

1. Our corpus-based study (described in Chapter 4) has shown that a surprisingly small set of deduction rules can cover most of the subsumption entailment-justification

pairs found in the corpus.

2. Our study on comparing candidate verbalisations of deduction rules (described in Chapter 6) indicates that for most of our rules, difficulty of inferences depends more on the deduction pattern than on which of several reasonable verbalisations is used for axioms.

3. Our study on the understandability of deduction rules (described in Chapter 7) has shown that even though our rules are all simple in the sense of having 1-3 premises, their understandability for human subjects covers a full range from 4% (very difficult) to 100% (very easy) success in recognising validity. Overall, the subjects found it difficult to understand inference steps that conclude an unsatisfiability. This suggests that it really matters *which* rules are used.

4. Our study on strategy-based explanations (described in Chapter 8) has shown that it is hard to find special strategies to elucidate difficult rules. Most strategies that we tried had little or no effect, the exceptions being the explicit statement of presuppositions and contextualisation of **Invs** statements.

11.3 Limitations of the Research

The main limitations of this research are as follows:

Rule coverage We did not try to collect deduction rules that can cover all justifications found in the corpus, but a small set of rules that can cover most of them. This approach is suitable for the practical limitations of a PhD project while still obtaining quite good coverage. Because of this, our system will, however, fail to produce a proof tree if it requires an inference step that is not covered by our rule set.

Lack of sophisticated planning Currently, we verbalise *all* inference steps in a proof tree to produce an explanation. This simple planning approach is unsuitable for complex proof trees as the presentation of too many (probably trivial) steps might hinder the readers' understanding of the overall inference. In such cases, the explanation should include only important steps.

Lack of test cases In evaluating the understandability model, we tested with only two-step inferences in OWL. It is unclear whether the model can generalise to OWL inferences with more than two steps.

Lack of testing with our target users in a realistic context In our studies, we tested with only ordinary users on CrowdFlower, but neither with domain experts nor knowledge engineers. Additionally, we used fictional words to theoretically test logical inferences in OWL (except in the preliminary evaluation study). In a realistic context, subjects might perform better [Was68, JLLL72].

Lack of deep data gathering methods Our preliminary evaluation study was conducted through questionnaires on CrowdFlower, and we did not ask the subjects to think aloud or debrief them after the test. Therefore, we do not know how they did the reasoning as well as utilized the explanations.

Lack of validation of final system Only a preliminary evaluation study was conducted.

Explanation generation In human-built ontologies, there are axioms that are too complex to be expressible in an unambiguous English sentence [PT10]. Identifying such axioms and methods to verbalise them is not a focus of our work. For this reason, our system will fail to produce an explanation if it contains such an axiom.

Proof tree construction Our proof tree construction algorithm also has a limitation in the sense that we did not try to find the best solution.

Most of the above-mentioned limitations are practical, and can be addressed in a larger project.

11.4 Broader Significance of the Work

1. We have proposed a novel method for measuring the cognitive difficulty of single-step inferences in OWL. Both the understandability indexes we have measured and the method for obtaining them might be useful in alternative models or contexts.

2. We have proposed a novel probabilistic model for predicting the understandability

of a multi-step inference based on those of single steps. This model can be used by other researchers to predict the understandability of different kinds of inferences.

3. In planning an explanatory text which contains multiple steps, the cognitive difficulty of individual steps should be taken into account. This is because their cognitive difficulty might vary greatly even though their structural complexity is simple (e.g., the trivial satisfaction of the universal restriction in OWL).

4. Our explanation tool can be integrated as a plug-in of ontology editing tools in order to help developers in inspecting undesired entailments, and potentially in debugging them (proving this is part of future work).

11.5 Future Work

Part of future work is to extend our current rule set to cover more inferences in OWL as well as measure their understandability. The findings and limitations of this thesis also suggest the following questions for further research in the future:

1. Are the explanations really helpful in debugging ontologies, and for which type of user (i.e., domain expert and/or ontology expert)?

2. Does the method for computing the understandability of a proof tree generalise to inferences with more than two steps?

3. How can we find a way to abstract a proof tree (e.g., grouping similar or related inference steps) in order to make important steps more visible to end-users? Additionally, how can we define different levels of abstraction and styles for an explanation in order to serve a wide variety of users?

4. For undesired entailments, how can we detect and highlight the main inference steps in a proof tree that lead to the absurd conclusion? And if possible, how can we provide suggestions for corrections in such cases?

5. How can we apply sophisticated NLG planning techniques (both macro-planning and micro-planning) to produce fluent and readable English explanations that can be understood easily by lay users?

Bibliography

[ABB+00] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, and Gerald M. Rubin. Gene Ontology: Tool for the Unification of Biology. *Nature Genetics*, 25(1):25–29, 2000.

[ADS+07] Harith Alani, David Dupplaw, John Sheridan, Kieron O'Hara, John Darlington, Nigel Shadbolt, and Carol Tullo. Unlocking the Potential of Public Sector Information with Semantic Web Technology. In *International/Asian Semantic Web Conference (ISWC/ASWC 2007)*, pages 708–721, 2007.

[AEC04] AECMA. AECMA Simplified Technical English, PSC-85-16598 "A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language". Technical report, The European Association of Aerospace Industries, 2004.

[AL98] John R. Anderson and Christian Lebiere. *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Inc., 1998.

[And80] Peter B. Andrews. Transforming Matings into Natural Deduction Proofs. In *International Conference on Automated Deduction (CADE 1980)*, pages 281–292, 1980.

[BBLPC] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. Turtle (Terse RDF Triple Language). `http://www.w3.org/TR/turtle/ /`. Last Accessed: 30th August 2010.

[BBMR89] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: a Structural Data Model for Objects. In *International Conference on Management of Data (SIGMOD 1989)*, pages 58–67, 1989.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[BCRM08] Alexander Borgida, Diego Calvanese, and Mariano Rodriguez-Muro. Explanation in the DL-Lite Family of Description Logics. In *Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE (OTM 2008)*, pages 1440–1457, 2008.

[Bea97] David Ian Beaver. *The Handbook of Logic and Language*, chapter 1, pages 939–1008. Elsevier, 1997.

[BFH+99] Alexander Borgida, Enrico Franconi, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. Explaining $\mathcal{ALC}$ Subsumption. In *International Workshop on Description Logics (DL 1999)*, pages 37–40, 1999.

[BFH00] Alex Borgida, Enrico Franconi, and Ian Horrocks. Explaining $\mathcal{ALC}$ Subsumption. In *European Conference on Artificial Intelligience (ECAI 2000)*, pages 209–213, 2000.

[BG01] Leo Bachmair and Harald Ganzinger. Resolution Theorem Proving. In *Handbook of Automated Reasoning*, pages 19–99. Elsevier and MIT Press, 2001.

[BH95] Franz Baader and Bernhard Hollunder. Embedding Defaults into Terminological Knowledge Representation Formalisms. *Automated Reasoning*, 14:149–180, 1995.

[BL84] Ronald J. Brachman and Hector J. Levesque. The Tractability of Subsumption in Frame-Based Description Languages. In *National Conference on Artificial Intelligence (AAAI 1984)*, 1984.

[Bli] Bliksem. `http://www.ii.uni.wroc.pl/~nivelle/software/bliksem/`. Last Accessed: 1st February 2013.

[BMPS+91] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori A. Resnick, Lori Alperin Resnick, and Alexander Borgida. Living with CLASSIC: When and How to Use a KL-ONE-Like Language. In *Principles of Semantic Networks: Explorations in the representation of knowledge*, pages 401–456, 1991.

[Bor92] Alexander Borgida. From Type Systems to Knowledge Representation: Natural Semantics Specifications for Description Logics. *Int. J. Cooperative Inf. Syst.*, 1(1):93–126, 1992.

[BPnS07] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the Description Logic $\mathcal{EL}^+$. In *German Conference on Advances in Artificial Intelligence (KI 2007)*, pages 52–67, 2007.

[BS99] Franz Baader and Ulrike Sattler. Expressive Number Restrictions in Description Logics. *Logic and Computation*, 9(3):319–350, 1999.

[BS01] Franz Baader and Ulrike Sattler. An Overview of Tableau Algorithms for Description Logics. *Studia Logica*, 69:5–40, 2001.

[BS08] Franz Baader and Boontawee Suntisrivaraporn. Debugging SNOMED CT Using Axiom Pinpointing in the Description Logic $\mathcal{EL}^+$. In *Knowledge Representation in Medicine Conference (KR-MED 2008)*, 2008.

[CDGL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lemho, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: tractable description logics for ontologies. In *National Conference on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[Che76] Daniel Chester. The translation of formal proofs into English. *Artificial Intelligence*, 7(3):261–278, 1976.

[Cos97] Yann Coscoy. A Natural Language Explanation for Formal Proofs. In *First International Conference on Logical Aspects of Computational Linguistics (LACL 1996)*, pages 149–167, 1997.

[CSM07] Anne Cregan, Rolf Schwitter, and Thomas Meyer. Sydney OWL Syntax - towards a Controlled Natural Language Syntax for OWL 1.1. In *International Workshop on OWL: Experiences and Directions (OWLED 2007)*, 2007.

[Dal92] Robert Dale. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes*. ACL-MIT Press, 1992.

[DFJ+04] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *International Conference on Information and Knowledge Management (CIKM 2004)*, pages 652–659, 2004.

[DIF+08] Brian Davis, Ahmad Ali Iqbal, Adam Funk, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham, and Siegfried Handschuh. RoundTrip Ontology Authoring. In *International Semantic Web Conference (ISWC 2008)*, pages 50–65, 2008.

[DMB+06] Martin Dzbor, Enrico Motta, Carlos Buil, Jose Gomez, Olaf Goerlitz, and Holger Lewen. Developing ontologies in OWL: An observational study. In *International Workshop on OWL: Experiences and Directions (OWLED 2006)*, 2006.

[EP93] Allan Edgar and Francis Pelletier. Natural language explanation of natural deduction proofs. In *Conference of the Pacific Association for Computational Linguistics*, 1993.

[Fie05] Armin Fiedler. Natural Language Proof Explanation. In *Mechanizing Mathematical Reasoning*, volume 2605, pages 342–363. Springer Berlin Heidelberg, 2005.

[Fri74] Michael Friedman. Explanation and Scientific Understanding. *Journal of Philosophy*, 71(1):5–19, 1974.

[FTB+07] Adam Funk, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham, Brian Davis, and Siegfried Handschuh. CLOnE: Controlled Language for Ontology Editing. In *International/Asian Semantic Web Conference (ISWC/ASWC 2007)*, pages 142–155, 2007.

[fuz] The fuzzyDL System. `http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html`. Last Accessed: 1st February 2013.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. II. *Mathematische Zeitschrift*, 39:405–431, 1935.

[GFH+03] Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The National Cancer Institute's Thesaurus and Ontology. *Journal of Web Semantics*, 1(1), 2003.

[GMT97] Vittorio Girotto, Alberto Mazzocco, and Alessandra Tasso. The effect of premise order in conditional reasoning: a test of the mental model theory. *Cognition*, 63:1–28, 1997.

[HB09] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for Working with OWL 2 Ontologies. In *International Workshop on OWL: Experiences and Directions (OWLED 2009)*, Virginia, USA, 2009.

[HBPS11] Matthew Horridge, Samantha Bail, Bijan Parsia, and Ulrike Sattler. The Cognitive Complexity of OWL Justifications. In *International Semantic Web Conference (ISWC 2011)*, pages 241–256, 2011.

[HDG+06] Matthew Horridge, Nicholas Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai Wang. The Manchester OWL Syntax. In *International Workshop on OWL: Experiences and Directions (OWLED 2006)*, 2006.

[HF97] Xiaorong Huang and Armin Fiedler. Proof verbalization as an application of NLG. In *International Joint Conference on Artifical Intelligence (IJCAI 1997)*, pages 965–970, 1997.

[HJD08] Glen Hart, Martina Johnson, and Catherine Dolbear. Rabbit: Developing a Control Natural Language for Authoring Ontologies. In *European Semantic Web Conference (ESWC 2008)*, pages 348–360, 2008.

[HKKHW05] Daniel Hewlett, Aditya Kalyanpur, Vladamir Kovlovski, and Chris Halaschek-Wiener. Effective Natural Language Paraphrasing of Ontologies on the Semantic Web. In *End User Semantic Web Interaction Workshop*, 2005.

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In *International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67, 2006.

[HM00] Volker Haarslev and Ralf Möller. Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles. In *International Conference on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 273–284, 2000.

[HM01] Volker Haarslev and Ralf Möller. RACER System Description. In *International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083, pages 701–705, 2001.

[HMS04] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In *International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 152–162, 2004.

[Hor98] Helmut Horacek. Generating Inference-Rich Discourse through Revisions of RST-Trees. In *Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence (AAAI/IAAI 1989)*, pages 814–820, 1998.

[Hor99] Helmut Horacek. Presenting Proofs in a Human-Oriented Way. In *International Conference on Automated Deduction (CADE 1999)*, pages 142–156, 1999.

[Hor08] Ian Horrocks. Ontologies and the Semantic Web. *Communications of the ACM*, 51(12):58–67, 2008.

[Hor11] Matthew Horridge. *Justification Based Explanation in Ontologies*. PhD thesis, University of Manchester, 2011.

[Hov88] Eduard H. Hovy. *Generating Natural Language under Pragmatic Constraints*. L. Erlbaum Associates Inc., Hillsdale, New Jersey, USA, 1988.

[HPS08a] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explanation of OWL Entailments in Protégé 4. In *International Semantic Web Conference - Posters & Demos (ISWC 2008)*, 2008.

[HPS08b] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and Precise Justifications in OWL. In *International Semantic Web Conference (ISWC 2008)*, pages 323–338, 2008.

[HPS08c] Matthew Horridge and Peter F. Patel-Schneider. Manchester OWL Syntax for OWL 1.1. In *International Workshop on OWL: Experiences and Directions (OWLED 2008)*, 2008.

[HPS09a] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. From Justifications to Proofs for Entailments in OWL. In *International Workshop on OWL: Experiences and Directions (OWLED 2009)*, 2009.

[HPS09b] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Lemmas for Justifications in OWL. In *International Workshop on Description Logics (DL 2009)*, 2009.

[HPS10] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Justification Oriented Proofs in OWL. In *International Semantic Web Conference (ISWC 2010)*, pages 354–369, 2010.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[HS99] Ian Horrocks and Ulrike Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Logic and Computation*, 9(3):385–410, 1999.

[Hua94] Xiaorong Huang. *Human Oriented Proof Presentation: A Reconstructive Approach.* PhD thesis, The University of Saarbrücken, Germany, 1994.

[HWGP+06] Christian Halaschek-Wiener, Jennifer Golbeck, Bijan Parsia, Vladimir Kolovski, and Jim Hendler. Image Browsing and Natural Language Paraphrases of Semantic Web Annotations. In *International Workshop on Semantic Web Annotations for Multimedia (SWAMM 2006)*, 2006.

[JHQ+09] Qiu Ji, Peter Haase, Guilin Qi, Pascal Hitzler, and Steffen Stadtmüller. RaDON – Repair and Diagnosis in Ontology Networks. In *European Semantic Web Conference (ESWC 2009)*, pages 863–867, 2009.

[JKD06] Mustafa Jarrar, Maria Keet, and Paolo Dongilli. Multilingual verbalization of ORM conceptual models and axiomatized ontologies. Technical report, STARLab, Vrije Universiteit Brussel, 2006.

[JLB84] Phillip N. Johnson-Laird and Bruno G. Bara. Syllogistic Inference. *Cognition*, 16(1):1–61, 1984.

[JLB91] Philip N. Johnson-Laird and Ruth M. J. Byrne. *Deduction*. Lawrence Erlbaum, 1991.

[JLLL72] P. N. Johnson-Laird, Paolo Legrenzi, and Maria Sonino Legrenzi. Reasoning and a Sense of Reality. *British Journal of Psychology*, 63(3):395–400, 1972.

[JQH09] Qiu Ji, Guilin Qi, and Peter Haase. A Relevance-Directed Algorithm for Finding Justifications of DL Entailments. In *Asian Semantic Web Conference (ASWC 2009)*, pages 306–320, 2009.

[Kal06] Aditya Kalyanpur. *Debugging and repair of OWL ontologies*. PhD thesis, The University of Maryland, 2006.

[KAO13] KAON2. `http://kaon2.semanticweb.org/`, Last Accessed: 1st February 2013.

[KF07] Kaarel Kaljurand and Norbert Fuchs. Verbalizing OWL in Attempto Controlled English. In *International Workshop on OWL: Experiences and Directions (OWLED 2007)*, 2007.

[Kit85] Philip Kitcher. Two Approaches to Explanation. *Journal of Philosophy*, 82(11):632–639, 1985.

[Kit03] Richard I. Kittredge. *The Oxford Handbook of Computational Linguistics*, chapter Sublanguages and Controlled Languages, pages 430–447. Oxford University Press, 2003.

[KMR04] Holger Knublauch, Mark A. Musen, and Alan L. Rector. Editing Description Logic Ontologies with the Protégé OWL Plugin. In *International Workshop on Description Logics (DL 2004)*, 2004.

[KPHS07] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding All Justifications of OWL DL Entailments. In *International Semantic Web Conference (ISWC 2007)*, 2007.

[KPS+06] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James A. Hendler. Swoop: A Web Ontology Editing Browser. *Journal of Web Semantics*, 4:144–153, 2006.

[KPSH05] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging Unsatisfiable Classes in OWL Ontologies. *Journal of Web Semantics*, 3(4):268–293, 2005.

[Kuh10] Tobias Kuhn. *Controlled English for Knowledge Representation.* PhD thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, 2010.

[Kuh13] Tobias Kuhn. The Understandability of OWL Statements in Controlled English. *Semantic Web*, 4(1):101–115, 2013.

[Kwo05] Francis King Hei Kwong. Practical approach to explaining $\mathcal{ALC}$ subsumption. Technical report, University of Manchester, 2005.

[Lin90] Christoph Lingenfelder. *Transformation and Structuring of Computer Generated Proofs.* PhD thesis, Universität of Kaiserslautern, 1990.

[Lip01] Peter Lipton. *What Good is An Explanation?*, chapter Explanation: Theoretical Approaches and Applications, pages 43–59. Kluwer Academic, 2001.

[LN04] Thorsten Liebig and Olaf Noppens. OntoTrack: Combining browsing and editing with reasoning and explaining for OWL Lite ontologies. In *International Semantic Web Conference (ISWC 2004)*, pages 244–258, 2004.

[LSPV08] Joey Sik Chun Lam, Derek Sleeman, Jeff Z. Pan, and Wamberto Vasconcelos. A Fine-Grained Approach to Resolving Unsatisfiable Ontologies. *Journal of Data Semantics*, pages 62–95, 2008.

[MB95] Deborah L. McGuinness and Alexander T. Borgida. Explaining Subsumption in Description Logics. In *IJCAI 1995, International Joint Conference on Artificial Intelligence*, pages 816–821, 1995.

[McD81] David D. McDonald. Mumble: A flexible system for language production. In *International Joint Conference on Artificial Intelligence (IJCAI 1981)*, pages 1062–1062, 1981.

[McD83] David D. McDonald. Natural Language Generation as a Computational Problem: An Introduction. In *Computational Models of Discourse*, chapter 4, pages 209–265. Michael Brady and Robert Berwick, 1983.

[McG96] Deborah Louise McGuinness. *Explaining reasoning in description logics.* PhD thesis, The State University of New Jersey, US, 1996.

[MEH04] Nicola J. Morley, Jonathan S. T. Evans, and Simon J. Handley. Belief Bias and Figural Bias in Syllogistic Reasoning. *The Quarterly Journal of Experimental Psychology*, 57(4):666–692, 2004.

[Met92] Marie W. Meteer. *Expressibility and the Problem of Efficient Text Planning.* St. Martin's Press, 1992.

[Mil84] Dale Miller. Expansion Tree Proofs and Their Conversion to Natural Deduction Proofs. In *International Conference on Automated Deduction (CADE 1984)*, pages 375–393, 1984.

[Mit99] Teruko Mitamura. Controlled Language for Multilingual Machine Translation. In *Machine Translation Summit VII*, 1999.

[MLBP06] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Z. Pan. Finding Maximally Satisfiable Terminologies for the Description Logic $\mathcal{ALC}$. In *National Conference on Artificial Intelligence (AAAI 2006)*, pages 269–274, 2006.

[Mot09] Boris Motik. Resolution-Based Reasoning for Ontologies. In *Handbook on Ontologies*, 2009.

[MP89] Johanna D. Moore and Cécile L. Paris. Planning Text for Advisory Dialogues. In *Annual Meeting of the Association for Computational Linguistics*, pages 203–211, 1989.

[MSH07] Boris Motik, Rob Shearer, and Ian Horrocks. A Hypertableau Calculus for $\mathcal{SHIQ}$. In *International Workshop on Description Logics (DL 2007)*, pages 419–426, 2007.

[MSP] MSPASS. `http://www.cs.man.ac.uk/~schmidt/mspass/`. Last Accessed: 1st February 2013.

[NBH+06] Stephen E. Newstead, Peter Bradon, Simon J. Handley, Ian Dennis, and Jonathan St. B. T. Evans. Predicting the difficulty of complex logical reasoning problems. *Thinking & Reasoning*, 12:1:62–90, 2006.

[NSW+09] Natalya F. Noy, Nigam H. Shah, Patricia L. Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L. Rubin, Margaret-Anne Storey, Christopher G. Chute, and Mark A. Musen. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37, 2009.

[Onta] Ontology Design Patterns. `http://ontologydesignpatterns.org`. Last Accessed: 1st February 2013.

[Ontb] Ontology Design Patterns Repository. `http://ontologydesignpatterns.org/ont/`. Last Accessed: 1st February 2013.

[Ott] Otter and Mace2. `http://www.cs.unm.edu/~mccune/mace2/`. Last Accessed: 1st February 2013.

[OWL04a] OWL Web Ontology Language - Abstract Syntax. `http://www.w3.org/TR/owl-semantics/syntax.html`, 2004. Last Accessed: 1st February 2013.

[OWL04b] OWL Web Ontology Language Overview. `http://www.w3.org/TR/owl-features/`, 2004. Last Accessed: 1st February 2013.

[OWL12a] OWL 2 Web Ontology Language Document Overview (Second Edition). `http://www.w3.org/TR/owl2-overview/`, 2012. Last Accessed: 1st February 2013.

[OWL12b] OWL 2 Web Ontology Language Profiles. `http://www.w3.org/TR/owl2-profiles/`, 2012. Last Accessed: 1st February 2013.

[OWL12c] OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). `http://www.w3.org/TR/owl2-syntax/`, 2012. Last Accessed: 1st February 2013.

[Pfe87] Frank Pfenning. *Proof Transformation in Higher-Order Logic.* PhD thesis, Carnegie Mellon University, 1987.

[PG86] David A. Plaisted and Steven Greenbaum. A Structure-Preserving Clause Form Translation. *J. Symb. Comput.*, 2(3):293–304, 1986.

[Pow12] Richard Power. OWL Simplified English: A Finite-State Language for Ontology Editing. In *International Workshop on Controlled Natural Language (CNL 2012)*, pages 44–60, 2012.

[PSK05] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL Ontologies. In *International Conference on World Wide Web (WWW 2005)*, pages 633–640, 2005.

[PT10] Richard Power and Allan Third. Expressing OWL axioms by English sentences: dubious in theory, feasible in practice. In *International Conference on Computational Linguistics: Posters (COLING 2010)*, pages 1006–1013, 2010.

[Pul96] Stephen G. Pulman. Controlled Language for Knowledge Representation. In *International Workshop on Controlled Language Applications*, pages 233–242, 1996.

[RDF04] RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2004. Last Accessed: 1st February 2013.

[RDH+04] Alan Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In *International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004)*, pages 63–81, 2004.

[Rei87] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[RMKM08] Daniel L. Rubin, Dilvan A. Moreira, Pradip Kanjamala, and Mark A. Musen. BioPortal: A Web Portal to Biomedical Ontologies. In *AAAI Spring Symposium: Symbiotic Relationships between Semantic Web and Knowledge Engineering*, pages 74–77, 2008.

[Rud11] Sebastian Rudolph. Foundations of Description Logics. In *Reasoning Web. Semantic Technologies for the Web of Data*, volume 6848, pages 76–136. Springer Berlin Heidelberg, 2011.

[Sal84] Wesley C. Salmon. *Scientific Explanation and the Causal Structure of the World*. Princeton University Press, 1984.

[SB07] Edward J. N. Stupple and Linden J. Ball. Figural Effects in a Syllogistic Evaluation Paradigm: An Inspection-Time Analysis. *Experimental Psychology*, 54 (2):120–127, 2007.

[SC03] Stefan Schlobach and Ronald Cornet. Non-standard Reasoning Services for the Debugging of Description Logic Terminologies. In *International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 355–360, 2003.

[SCC97] Kent A. Spackman, Keith E. Campbell, and Roger A. Côté. SNOMED RT: A Reference Terminology for Health Care. In *AMIA Annual Fall Symposium*, pages 640–644, 1997.

[Sch88] Uwe Schöning. Graph isomorphism is in the low hierarchy. *J. Comput. Syst. Sci.*, 37(3):312–323, 1988.

[Sch10] Rolf Schwitter. Controlled Natural Languages for Knowledge Representation. In *International Conference on Computational Linguistics: Posters (COLING 2010)*, pages 1113–1121, 2010.

[SHCH07] Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank Harmelen. Debugging Incoherent Terminologies. *J. Autom. Reason.*, 39(3):317–349, 2007.

[SKC+08] Rolf Schwitter, Kaarel Kaljurand, Anne Cregan, Catherine Dolbear, and Glen Hart. A Comparison of three Controlled Natural Languages for OWL 1.1. In *International Workshop on OWL: Experiences and Directions (OWLED 2008)*, 2008.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A Practical OWL-DL Reasoner. *Journal of Web Semantics*, 5:51–53, 2007.

[SQJH08] Boontawee Suntisrivaraporn, Guilin Qi, Qiu Ji, and Peter Haase. A Modularization-Based Approach to Finding All Justifications for OWL DL Entailments. In *Asian Semantic Web Conference (ASWC 2008)*, 2008.

[SS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[ST04] Rolf Schwitter and Marc Tilbrook. Controlled Natural Language meets the Semantic Web. In *Australasian Language Technology Workshop (ALTW 2004)*, pages 55–62, 2004.

[Stu08] Heiner Stuckenschmidt. Debugging OWL Ontologies – A Reality Check. In *International Workshop on Evaluation of Ontology-based Tools and the Semantic Web Service Challenge (EON-SWSC 2008)*, 2008.

[Sun09] Boontawee Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. PhD thesis, Technical University of Dresden, 2009.

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *International Joint Conference on Automated Reasoning (IJCAR 2006)*, pages 292–297, 2006.

[TON] The TONES Ontology Repository. `http://owl.cs.manchester.ac.uk/repository/`. Last Accessed: 1st February 2013.

[Top] TopQuadrant. TopBraid Composer. `http://www.topquadrant.com/products/TB_Composer.html`. Last Accessed: 1st February 2013.

[Was68] Peter Cathcart Wason. Reasoning about a Rule. *Quarterly Journal of Experimental Psychology*, 20(3):273–281, 1968.

[WDF+09] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. SPASS Version 3.5. In *International Conference on Automated Deduction (CADE 2009)*, pages 140–145, 2009.

[WO98] Huijsen Willem-Olaf. Controlled Language—An Introduction. In *International Workshop on Controlled Language Applications (CLAW 1998)*, pages 1–15, 1998.

[WT92] Michael R. Wick and William B. Thompson. Reconstructive expert system explanation. *Artificial Intelligence*, 54:33–70, 1992.

[WTP11] Sandra Williams, Allan Third, and Richard Power. Levels of Organisation in Ontology verbalisation. In *European Workshop on Natural Language Generation (ENLG 2011)*, pages 158–163, 2011.

Appendix A

Frequently Occurring Deduction Patterns

Table A.1 lists all nine deduction patterns for $\top \sqsubseteq A$ entailments, and ten frequently occurring deduction patterns for $A \sqsubseteq \bot$ and $A \sqsubseteq B$ entailments (where A and B are class names) that were collected from the empirical study described in Chapter 4.

Table A.1: Frequently occurring deduction patterns for each entailment type sorted by ontology frequency ('O') then justification frequency ('J')

Entailment	Justification	O	J
$\top \sqsubseteq C_0$	1. $C_0 \equiv C_1 \sqcup C_2$ 2. $C_1 \equiv \neg C_2$	3 (50.0%)	6 (21.4%)
	1. $C_1 \sqsubseteq C_0$ 2. $C_2 \sqsubseteq C_0$ 3. $C_1 \equiv \neg C_2$	3 (50.0%)	6 (21.4%)
	1. $\mathbf{Dom}(r_0, C_0)$ 2. $C_1 \equiv \forall r_0 C_2$ 3. $C_1 \sqsubseteq C_3$ 4. $C_3 \sqsubseteq C_0$	2 (33.3%)	3 (10.7%)
	1. $\mathbf{Dom}(r_0, C_0)$ 2. $C_1 \equiv (\forall r_0 C_2) \sqcup (\exists r_1 C_3)$ 3. $C_1 \sqsubseteq C_4$ 4. $C_4 \sqsubseteq C_0$	2 (33.3%)	3 (10.7%)
	1. $\mathbf{Dom}(r_0, C_0)$ 2. $C_1 \equiv \forall r_0 (C_2 \sqcup C_3)$ 3. $C_1 \sqsubseteq C_4$ 4. $C_4 \sqsubseteq C_0$	1 (16.7%)	3 (10.7%)
	1. $\mathbf{Dom}(r_0, C_0)$ 2. $C_1 \equiv (\forall r_0 C_2) \sqcup (\exists r_0 C_3)$ 3. $C_1 \sqsubseteq C_0$	1 (16.7%)	2 (7.1%)
	1. $\mathbf{Dis}(C_1, C_2)$ 2. $\mathbf{Dom}(r_0, C_1)$ 3. $C_4 \equiv \forall r_0 C_5$ 4. $C_4 \sqsubseteq C_0$ 5. $C_0 \sqsubseteq C_1$ 6. $C_3 \sqsubseteq C_2$ 7. $\mathbf{Rng}(r_0, C_3)$	1 (16.7%)	2 (7.1%)
	1. $\mathbf{Dis}(C_1, C_2)$ 2. $\mathbf{Dom}(r_0, C_1)$ 3. $r_1 \sqsubseteq r_0$ 4. $C_4 \equiv \forall r_1 C_5$ 5. $C_4 \sqsubseteq C_0$ 6. $C_0 \sqsubseteq C_1$ 7. $C_3 \sqsubseteq C_2$ 8. $\mathbf{Rng}(r_1, C_3)$	1 (16.7%)	2 (7.1%)
	1. $\mathbf{Dom}(r_0, C_0)$ 2. $r_1 \sqsubseteq r_0$ 3. $C_1 \equiv \forall r_1 C_2$ 4. $C_1 \sqsubseteq C_3$ 5. $C_3 \sqsubseteq C_0$	1 (16.7%)	1 (3.6%)
$C_0 \sqsubseteq \bot$	1. $C_0 \sqsubseteq C_1$ 2. $C_0 \sqsubseteq C_2$ 3. $C_2 \sqsubseteq C_3$ 4. $C_3 \sqsubseteq C_4$ 5. $\mathbf{Dis}(C_1, C_4)$	10 (31.3%)	16 (0.8%)
	1. $C_0 \sqsubseteq C_1$ 2. $C_0 \sqsubseteq C_3$ 3. $C_1 \sqsubseteq C_2$ 4. $C_3 \sqsubseteq C_4$ 5. $\mathbf{Dis}(C_2, C_4)$	8 (25.0%)	15 (0.8%)
	1. $C_0 \sqsubseteq C_1$ 2. $C_0 \sqsubseteq C_2$ 3. $C_2 \sqsubseteq C_3$ 4. $\mathbf{Dis}(C_1, C_3)$	6 (18.8%)	38 (2.0%)

Continued on Next Page...

	1. $C_1 \equiv C_0 \sqcup C_2 \sqcup C_3$ 2. $C_1 \sqsubseteq C_4$ 3. $C_4 \sqsubseteq C_5$ 4. **Dis**(C_0, C_5)	2 (6.3%)	5 (0.3%)
	1. $C_0 \equiv C_3 \sqcap \exists r_0.C_4 \sqcap = nr_1.T$ 2. $C_1 \equiv C_3 \sqcap \exists r_0.C_4 \sqcap = nr_1.T$ 3. $C_2 \equiv C_3 \sqcap \exists r_0.C_4 \sqcap = nr_1.T$ 4. **Dis**(C_1, C_2)	1 (3.1%)	96 (5.0%)
	1. $C_0 \sqsubseteq C_1$ 2. $C_1 \sqsubseteq \exists r_0.C_2$ 3. **Dom**(r_0, C_3) 4. $C_3 \sqsubseteq \exists r_1.C_4$ 5. **Rng**(r_1, C_5) 6. **Dis**(C_4, C_5)	1 (3.1%)	19 (1.0%)
	1. $C_0 \sqsubseteq C_1$ 2. $C_1 \sqsubseteq C_2$ 3. **Dom**(r_0, C_3) 4. $r_1 \sqsubseteq r_0$ 5. $C_4 \equiv \forall r_1.C_5$ 6. $C_4 \sqsubseteq C_6$ 7. $C_6 \sqsubseteq C_3$ 8. **Dis**(C_2, C_3)	1 (3.1%)	16 (0.8%)
	1. $C_0 \sqsubseteq C_1$ 2. $C_1 \sqsubseteq C_2$ 3. **Dom**(r_0, C_3) 5. **Invs**(r_0, r_1) 4. $C_2 \sqsubseteq \exists r_1.C_4$ 6. **Dis**(C_3, C_4)	1 (3.1%)	12 (0.6%)
	1. $C_0 \sqsubseteq C_1$ 2. $C_1 \sqsubseteq C_2 \sqcap \exists r_0.(C_3 \sqcup C_4)$ 3. $C_2 \sqsubseteq C_5 \sqcap \exists r_1.C_6 \sqcap \forall r_1.C_6$ 4. $C_6 \equiv C_7$ 5. $C_7 \equiv C_8$ 6. $C_8 \sqsubseteq C_{10} \sqcap \exists r_2.C_{11} \sqcap \forall r_2.C_{11}$ 7. $C_9 \sqsubseteq C_{10} \sqcap \exists r_2.C_{12} \sqcap \forall r_2.C_{12}$ 8. $C_{11} \sqsubseteq C_{12}$ 9. **Dis**(C_6, C_9)	1 (3.1%)	11 (0.6%)
	1. $C_0 \sqsubseteq C_1$ 2. $C_1 \equiv C_2 \sqcap \exists r_0.C_3 \sqcap \exists r_0.C_4 \sqcap \exists r_0.C_5 \sqcap \exists r_1.C_6$ 3. $C_3 \equiv C_7 \sqcap \exists r_2.(C_8 \sqcap \exists d_0.\{l_0 \star D_{t0}\})$ 4. **Rng**(d_0, D_{t1}), D_{t0} & D_{t1} are disjoint	1 (3.1%)	9 (0.5%)
$C_0 \sqsubseteq C_1$	1. $C_0 \sqsubseteq C_2$ 2. $C_2 \equiv C_1$	47 (28.1%)	3,204 (2.1%)
	1. $C_0 \equiv C_2$ 2. $C_2 \sqsubseteq C_1$	47 (28.1%)	1,804 (1.2%)
	1. $C_0 \sqsubseteq C_2$ 2. $C_2 \sqsubseteq C_1$	43 (25.7%)	939 (0.6%)
	1. $C_0 \equiv C_1 \sqcap \exists r_0.C_2$	35 (21.0%)	895 (0.6%)
	1. $C_0 \sqsubseteq = nd_0.D_{t0}$, $n > 0$ 2. **Dom**(d_0, C_1)	22 (13.2%)	50 (0.0%)
	1. $C_0 \sqsubseteq C_2$ 2. $C_2 \sqsubseteq \leq nr_0.\top$, $n > 0$ 3. $C_2 \sqsubseteq C_3$ 4. $C_3 \sqsubseteq C_4$ 5. **Invs**(r_0, r_1) 6. $r_1 \sqsubseteq r_2$ 7. **Rng**$(r_2, C_1 \sqcup C_5)$ 8. $C_5 \equiv (C_6 \sqcup (C_7 \sqcap \exists r_3.C_6)) \sqcap \exists r_4.C_2$	15 (9.0%)	140 (0.1%)

Continued on Next Page...

	9. $\mathbf{Dis}(C_4, C_7)$ 10. $\mathbf{Dis}(C_3, C_6)$		
	1. $C_0 \sqsubseteq \exists r_0.C_2$ 2. $C_1 \equiv \exists r_4.C_3$ 3. $\mathbf{Invs}(r_0, r_1)$ 4. $r_1 \sqsubseteq r_2$ 5. $\mathbf{Invs}(r_2, r_3)$ 6. $r_3 \sqsubseteq r_4$ 7. $\mathbf{Rng}(r_0, C_3)$	15 (9.0%)	223 (0.1%)
	1. $C_0 \sqsubseteq C_2$ 2. $C_2 \sqsubseteq \geq n r_0.\top$, $n > 0$ 3. $\mathbf{Rng}(r_0, C_3)$ 4. $C_3 \equiv C_4$ 5. $C_1 \equiv (\exists r_0.C_4) \sqcup (\exists r_1.C_5)$	14 (8.4%)	79 (0.1%)
	1. $C_0 \sqsubseteq \exists d_0.\{l_0 \star D_{t0}\}$ 2. $\mathbf{Dom}(d_0, C_1)$	9 (5.4%)	2,569 (1.7%)
	1. $C_0 \sqsubseteq \exists r_0.(C_2 \sqcap \exists r_0.C_3)$ 2. $C_0 \sqsubseteq C_4$ 3. $C_3 \sqsubseteq C_4$ 4. $\mathbf{Tra}(r_0)$ 5. $C_1 \equiv C_4 \sqcap \exists r_0.C_4$	5 (3.0%)	168 (0.1%)

Appendix B

Subjects' Comments from the Preliminary Evaluation Study

This chapter reports the subjects' comments for each question in the preliminary evaluation study. Some of these comments compare explanations (e.g., proof tree versus text); others describe the subjects' difficulties when they were doing the task. Such comments provide good suggestions for deciding how the final explanation is best set up as well as designing a more extensive evaluation study of the tool in the future. The corrections proposed by the associated subjects are also presented here.

QUESTION 1.1 (PATTERN 1 EXPLANATION 1)

EXPLANATION:

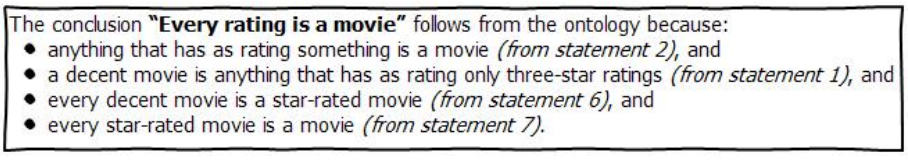


Figure B.1: The explanation in Question 1.1 (pattern 1 explanation 1), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: None of the statements

Correction: Every rating is a movie is a correct statement. It does not need to be corrected.

Comment: The explanation is longer than it needs to be. The second statement "Anything that has a rating something is a movie" draws the conclusion any rating is a movie.

2. *Problematic statement*: None of the statements

   Correction: None

   Comment: I do not agree with "Every rating is a movie" would be outputted. Since only objects with ratings would be considered movies, and ratings do not have ratings of themselves. The object would be the unique identifier in a database, not the ratings themselves. So these two entities could not be linked, and assumes one is another. Since they do not have common class that could be given the title of movie.

3. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: Anything that has rating something is not a movie.

   Comment: The phrase "anything that has rating something" is confused as it never occurs in real life.

4. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: Every movie has a rating.

   Comment: I don't know OWL.

QUESTION 2.1 (PATTERN 2 EXPLANATION 1)

EXPLANATION:

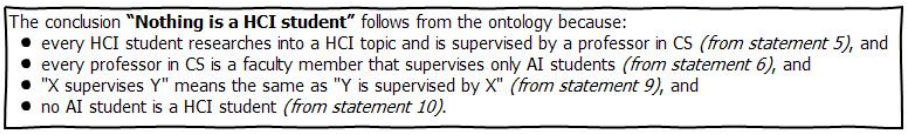


Figure B.2: The explanation in Question 2.1 (pattern 2 explanation 1), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Every professor in CS is a faculty member that supervises only AI

students. This statement is incorrect. The next statement clearly shows that HCI Student Researchers are supervised by professors in CS. It should say, a CS professor supervises either HCI students or AI students, but not both. Every HCI student researches into a HCI topic and is supervised by a professor in CS.

Comment: Yes, this is very difficult task.

QUESTION 3.1 (PATTERN 3 EXPLANATION 1)

EXPLANATION:

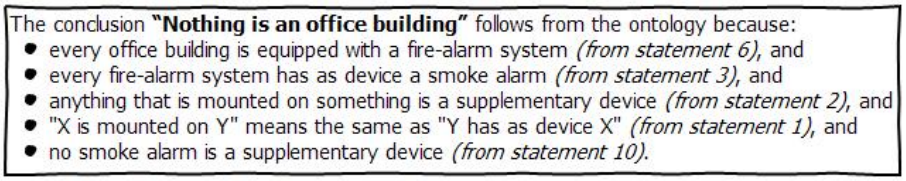


Figure B.3: The explanation in Question 3.1 (pattern 3 explanation 1), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: A smoke alarm is a supplementary device.

   Comment: It looks to me that since it isn't even true that a smoke alarm could not be a supplementary device at the same time it COULD be mounted.

2. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: Statement 10, that no smoke alarm is a supplementary device, is contradictory to previous statements, particularly statements 2 and 3. By simply removing this statement, the other statements are acceptable.

   Comment: This was not confusing; I love logic games like this!

3. *Problematic statement*: None of the statements

   Correction: I'm not sure how the computer drew the conclusion, so I don't know which statement to remove.

   Comment: I just couldn't follow the long list of statements that the computer used to draw the conclusion.

QUESTION 4.1 (PATTERN 4 EXPLANATION 1)

EXPLANATION:

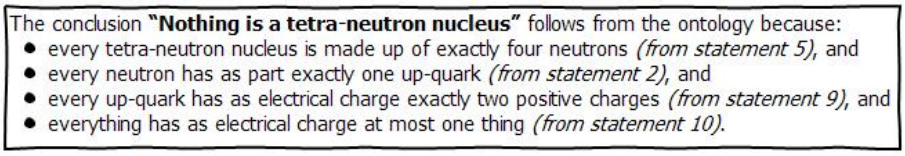


Figure B.4: The explanation in Question 4.1 (pattern 4 explanation 1), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Everything has as electrical charge at least one thing.

   Comment: I presume one of the statements is inaccurate, and I lack the experience to know which.

2. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Statement 10 is worded very oddly in general, so my best guess is that it would need changing, but I am not sure what an improvement would be without better understanding what it's intended to mean.

   Comment: None

3. *Problematic statement*: None of the statements

   Correction: None

   Comment: The concept of neutrons and quarks is very confusing.

QUESTION 5.1 (PATTERN 5 EXPLANATION 1)

EXPLANATION:

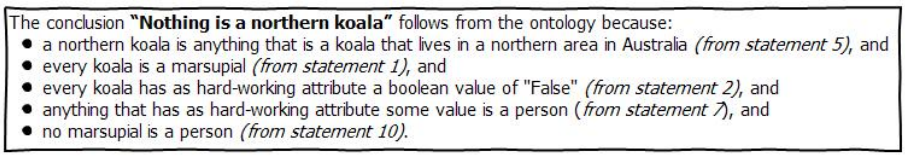


Figure B.5: The explanation in Question 5.1 (pattern 5 explanation 1), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that has as hard-working attribute some value is a person.

Correction: I'm confused how to fix the result without leading to another incorrect result (quokkas don't exist). To fix both, either Statement 7 and 4 would have to be changed, or 6 and 7 to something besides Boolean. I don't know whether I didn't make the correct change or I'm just meant to ignore quokkas because I wasn't asked about them.

Comment: None

QUESTION 1.2 (PATTERN 1 EXPLANATION 2)

EXPLANATION:

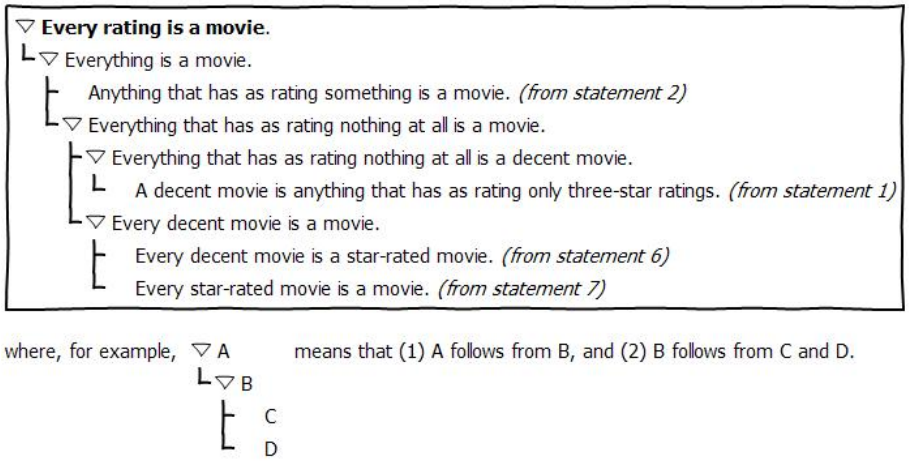


Figure B.6: The explanation in Question 1.2 (pattern 1 explanation 2), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: I would remove Statement 2, as it is redundant and illogical due to Statement 7.

   Comment: This was a bit confusing, due to having to really think like a computer program.

2. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: Anything that is star rated and has rating from 0 to 5 is a movie. If nothing at all is something then the requirement needs to be more specific.

   Comment: None

3. *Problematic statement*: None of the statements

   Correction: None

Comment: I don't see why "everything has rating nothing at all" follows from statement 1. This seems an incorrect inference. Perhaps the program misinterprets the (bad) English of statement 1.

QUESTION 2.2 (PATTERN 2 EXPLANATION 2)

EXPLANATION:

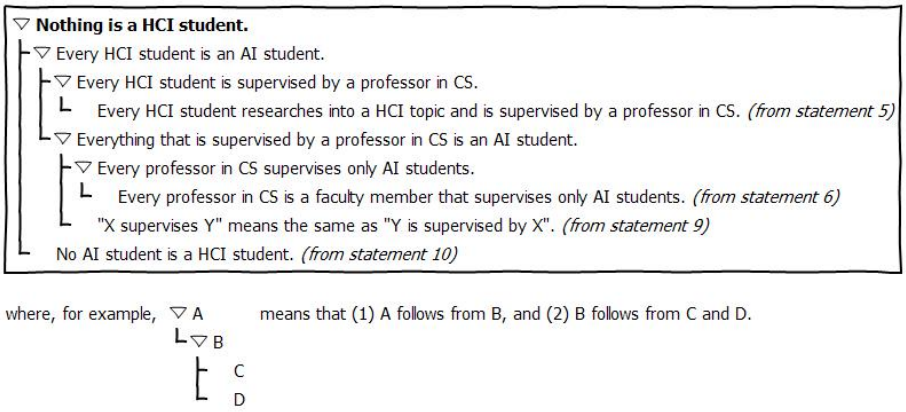


Figure B.7: The explanation in Question 2.2 (pattern 2 explanation 2), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

No comments provided.

QUESTION 3.2 (PATTERN 3 EXPLANATION 2)

EXPLANATION:

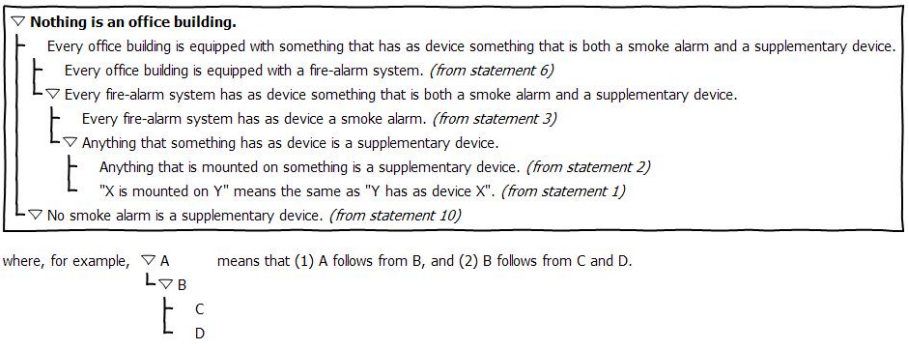


Figure B.8: The explanation in Question 3.2 (pattern 3 explanation 2), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that is mounted on something is a supplementary device.

Correction: Some things that are mounted on something are supplementary devices.

Comment: It was very confusing.

QUESTION 4.2 (PATTERN 4 EXPLANATION 2)

EXPLANATION:

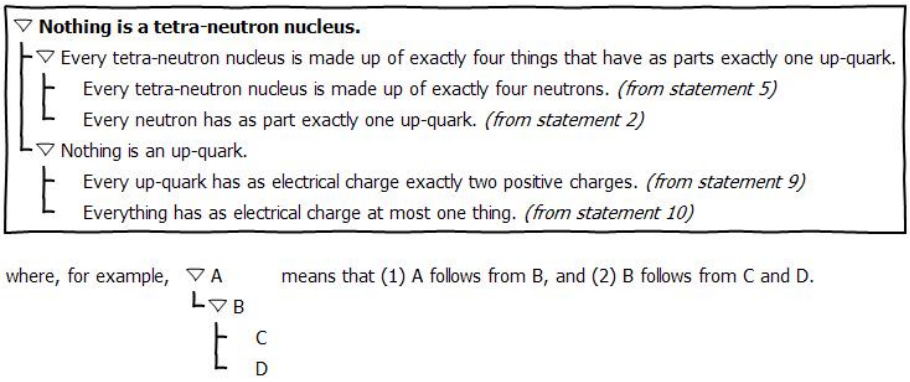


Figure B.9: The explanation in Question 4.2 (pattern 4 explanation 2), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: I would change the statement to read "Everything has electrical charge at least one thing".

   Comment: Up-quarks, tetra-nucleons, etc. are highly scientific terms that are hard for lay people to conceptualize and attach significance to.

2. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: It could be stricken entirely or changed to read "Everything has as electrical charge at least one thing".

   Comment: It was hard to see how many possible charges could make a nucleus.

3. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Everything has as an electrical charge at least two things.

   Comment: There were multiple possible changes that could change the conclusion.

4. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Either S9 or S10 should be corrected, depending on which is untrue. Either up-quarks do not have more than one positive charge, or some things can

have more than one electrical charge.

Comment: This second explanation shows the hierarchy of reasoning much more clearly.

QUESTION 5.2 (PATTERN 5 EXPLANATION 2)

EXPLANATION:

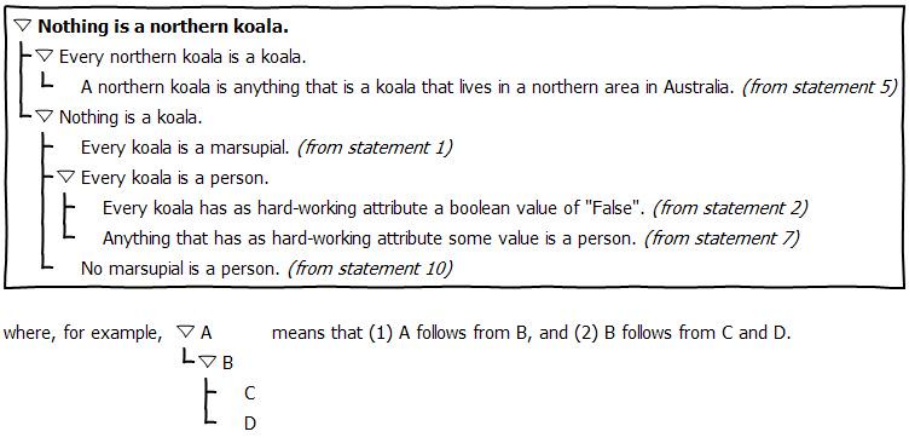


Figure B.10: The explanation in Question 5.2 (pattern 5 explanation 2), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that has as hard-working attribute some value is a person.

   Correction: I would either omit Statement 7 or change to "Anything that has a hard-working attribute some value is not a person".

   Comment: Statement 5 is difficult to read. I think it should be written as follows: A northern koala is anything that is a koala and lives in northern Australia.

2. *Problematic statement*: Anything that has as hard-working attribute some value is a person.

   Correction: Change to "Anything that has as hard-working attribute some value is an animal".

   Comment: I found the text of the hard-working attributes statements very difficult to process—they don't make a whole lot of sense.

QUESTION 1.3 (PATTERN 1 EXPLANATION 3)

EXPLANATION:

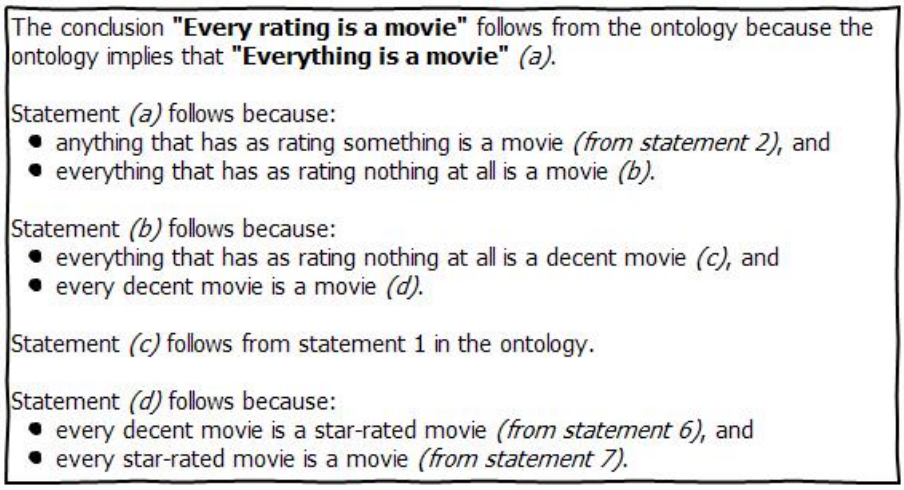


Figure B.11: The explanation in Question 1.3 (pattern 1 explanation 3), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: The statement, "Anything that has as rating something is a movie", would need to be removed.

   Comment: The statement, "Anything that has as rating something is a movie", is grammatically confusing.

2. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: I would change Statement 2 to: Anything that has a rating something is a movie. I think the computer is thinking "as rating" is an independent thing.

   Comment: None

3. *Problematic statement*: None of the statements

   Correction: None

   Comment: Even though the explanation was a bit difficult to follow I could understand how a computer could come to the answer.

4. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: Some things that have a rating are movies.

   Comment: Too many logic statements

5. *Problematic statement*: Anything that has as rating something is a movie.

Correction: To avoid arriving at the conclusion that "Every rating is a movie" requires the omission of statement 2.

Comment: I found the above questions difficult because they didn't appear to be grammatically correct for my understanding of English usage. I believe the explanation is deriving incorrect conclusions at various points such as (b) and (c).

QUESTION 2.3 (PATTERN 2 EXPLANATION 3)

EXPLANATION:

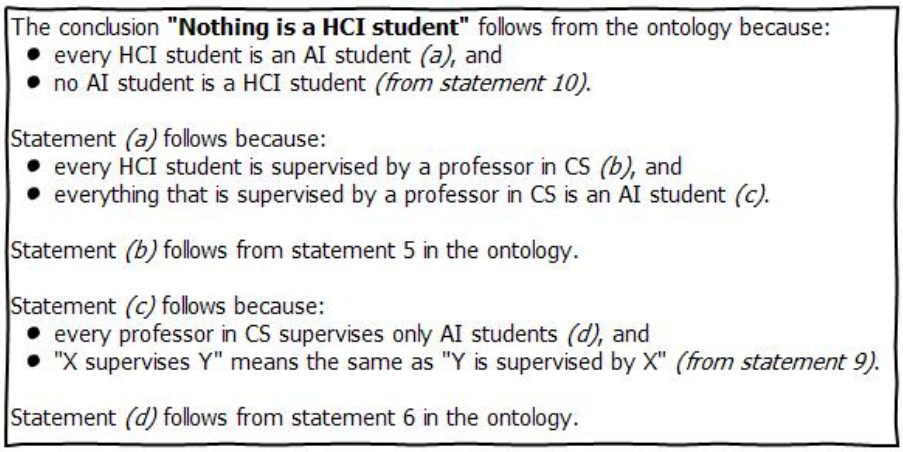


Figure B.12: The explanation in Question 2.3 (pattern 2 explanation 3), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Either omit "AI", or change to "AI or HCI", or omit "only". Other possibility is to remove statement 10 or change it to "Every HCI and AI student is a HCI student and an AI student".

   Comment: Nope, but you might want phrase the computer conclusion in more common language, as for some people "Nothing is..." might sound confusing. I would explain that this statement means "There are no HCI students".

2. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Statement 6 should read "Every professor in CS is a faculty member that supervises AI and/or HCI students".

Comment: The explanation did eventually make sense after following every step. For me, an easier explanation would be that the combination of Statement 6 and Statement 5 means that an HCI student cannot be supervised by a CS professor unless they are also a AI student, which Statement 10 rules out.

3. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Statement 5 and statement 6 are contradictory. In statement 5 it states, "Every HCI student... is supervised by a professor in CS". In statement 6, it states, "Every professor in CS...supervises only AI students". To avoid confusion and the conclusion "Nothing is an HCI student", I would change the end of statement 6 to "... that supervises only AI AND HCI students".

   Comment: It took me a minute to figure out the explanation. I had to think about what it was saying, making slightly harder to understand.

QUESTION 3.3 (PATTERN 3 EXPLANATION 3)

EXPLANATION:

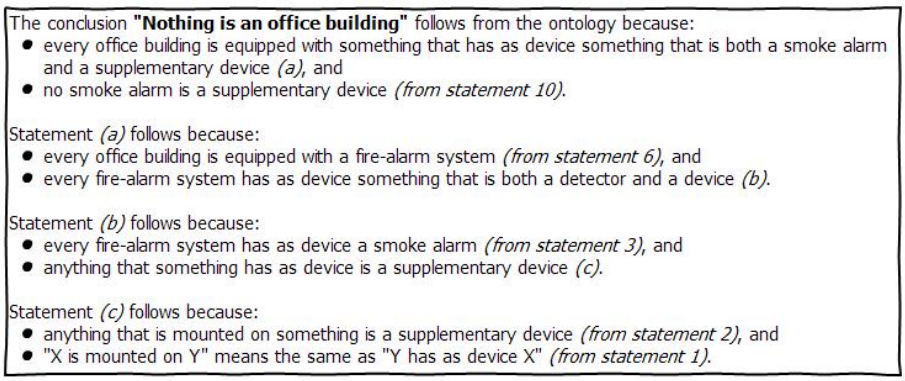


Figure B.13: The explanation in Question 3.3 (pattern 3 explanation 3), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Every fire-alarm system has as device a smoke alarm.

   Correction: If anything that is mounted is a supplementary device and no smoke alarms are supplementary devices, then (per Statement 6) no building with a smoke alarm is an office building. If statement 3 is changed to "Some fire-alarm systems do not have smoke alarms" then some buildings could, hypothetically, be office build-

ings. I think.

Comment: They are very narrow in their scope and the logical reasoning is difficult between the different statements. They lead to circular thinking.

2. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: A smoke alarm is a supplementary device.

   Comment: Extra statements that aren't part of the proof, and statements are out of the order of the proof making it more difficult to follow.

3. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: Smoke alarm is a supplementary device.

   Comment: The phrase "something that has as device something" is very confused.

QUESTION 4.3 (PATTERN 4 EXPLANATION 3)

EXPLANATION:

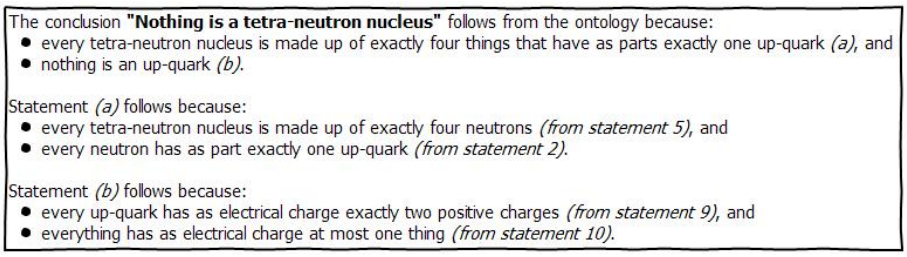


Figure B.14: The explanation in Question 4.3 (pattern 4 explanation 3), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Every up-quark has as electrical charge exactly two positive charges.

   Correction: I see a conflict between 9 and 10, but I don't know what is an up-quark so I don't know what would be the correct modification.

   Comment: Lack of knowledge on the semantic domain makes it more difficult to follow. On the other hand, familiarity is also dangerous because one may use "known" statements that are not in the ontology.

2. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: None

Comment: I do not understand the sentence "Everything has as electrical charge at most one thing".

QUESTION 5.3 (PATTERN 5 EXPLANATION 3)

EXPLANATION:

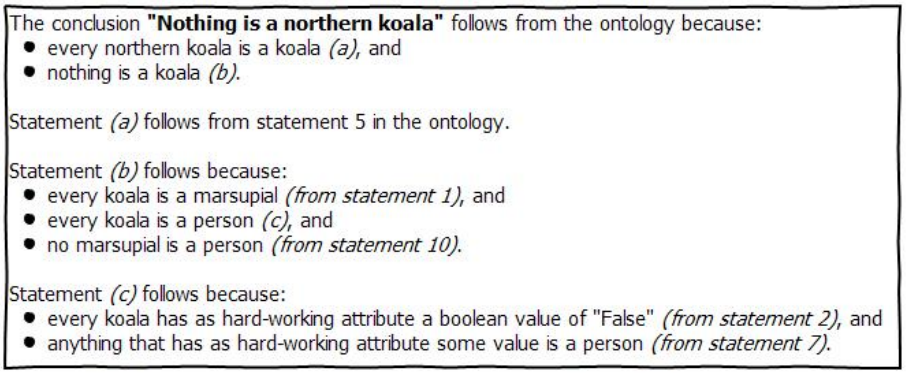


Figure B.15: The explanation in Question 5.3 (pattern 5 explanation 3), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that has as hard-working attribute some value is a person.

   Correction: Every person has hard-working as an attribute some value.

   Comment: There are multiple statements and some of them aren't really true.

2. *Problematic statement*: Anything that has as hard-working attribute some value is a person.

   Correction: I would change it to read "Anything that has as hard-working attribute is an animal".

   Comment: I think the terminology and phrasing is in correct grammatically for some of these sentences and the aspects of hard-working and boolean values muddied the other statements.

3. *Problematic statement*: None of the statements

   Correction: None

   Comment: This explanation was much more confusing to follow, I liked the flow chart of the previous explanation much better.

4. *Problematic statement*: None of the statements

   Correction: There is no reason for the computer to make that conclusion based on the statements given.

   Comment: None.

QUESTION 1.4 (PATTERN 1 EXPLANATION 4)

EXPLANATION:

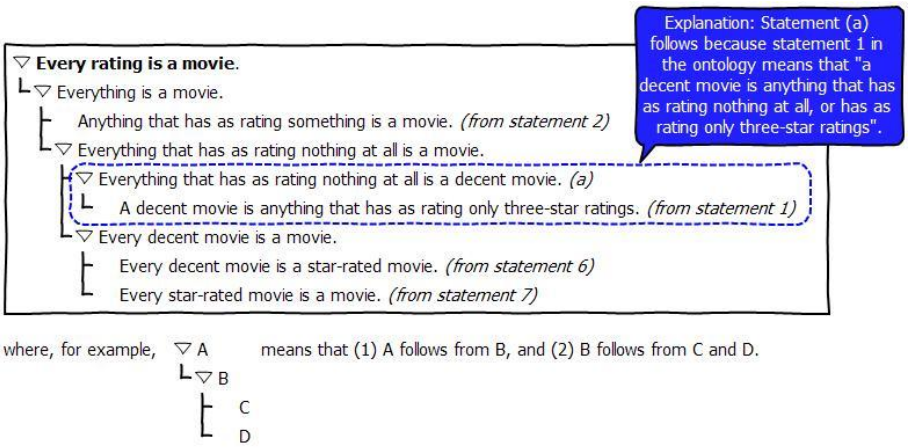


Figure B.16: The explanation in Question 1.4 (pattern 1 explanation 4), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: A decent movie is anything that has as rating only three-star ratings.

   Correction: A decent movie is any movie that has as rating only three-star ratings.

   Comment: Not sure I am right on this one. I don't follow the logic that "A decent movie is anything that has as rating only three-star ratings" allows you to state "Everything that has as rating nothing at all is a decent movie".

2. *Problematic statement*: None of the statements

   Correction: Statement (a) is contradictory to Statement 1. It basically saying "If I have either no rating or a 3 star rating, then it is a decent movie".

   Comment: None

3. *Problematic statement*: None of the statements

   Correction: I'm really not sure how to fix it.

Comment: I just can't follow the statements to figure out why the computer concluded every rating is a movie.

4. *Problematic statement*: None of the statements

   Correction: None

   Comment: I don't understand why Statement 1 "A decent movie is anything that has as rating only three-star ratings" means "a decent movie is anything that has as rating nothing at all, or has as rating only three-star ratings".

5. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: Every movie has a rating of something.

   Comment: Slightly confusing because there are a lot of similar statements with slightly differed words.

QUESTION 2.4 (PATTERN 2 EXPLANATION 4)

EXPLANATION:

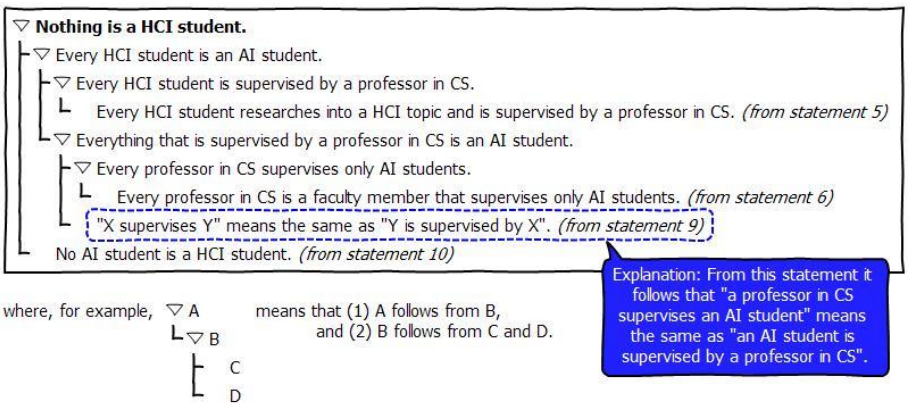


Figure B.17: The explanation in Question 2.4 (pattern 2 explanation 4), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Well, if Statement 6 stated that CS professors supervised both AI and HCI students, the conclusion in Statement 10 would no longer follow.

   Comment: From reading the ten statements, I, who am not a computer, concluded that Statement 10 followed from Statements 5 and 6. However, your explanation

and chart only confused me. Note, I did study symbolic logic circa 1980, but not computer science.

2. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Every professor in CS is a faculty member that supervises only AI students or HCI students.

   Comment: Logic is hard.

QUESTION 3.4 (PATTERN 3 EXPLANATION 4)

EXPLANATION:

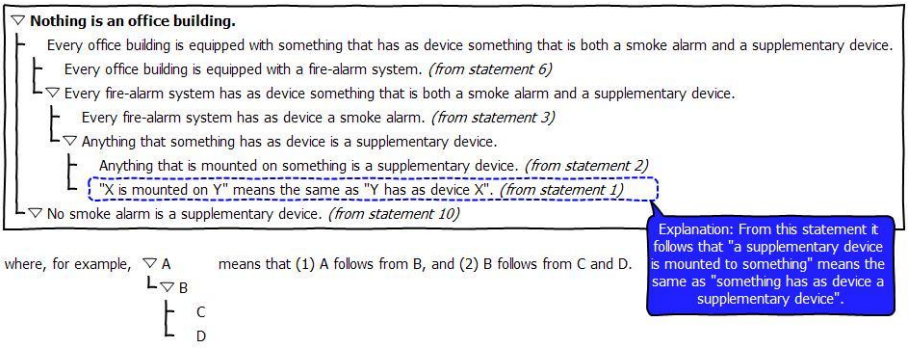


Figure B.18: The explanation in Question 3.4 (pattern 3 explanation 4), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that is mounted on something is a supplementary device.

   Correction: Not everything that is mounted on something is a supplementary device. If I am riding or "mounted on" a bicycle, the bicycle is the supplementary device.

   Comment: None

2. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: Statement 10 contradicts the first three statements...so you have to choose which one you'd like to keep.

   Comment: Poor diagram

3. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: Must be deleted, or could be corrected in many ways, e.g., A smoke

alarm may not be a supplementary device.

Comment: Hard to keep track of the multiple relevant statements, esp. given the inverse relationship between mounted on and has as device.

4. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: Every smoke alarm is a supplementary device. Smoke alarm is mounted on every fire alarm system. Every life protection system is a fire alarm system therefore has a smoke alarm mounted on it. Every office building is equipped with a fire alarm system. Because the smoke alarm is mounted on the fire alarm system it is a supplementary device. "No smoke alarm is a supplementary device" contradicts previous statement and can't be mounted on the fire alarm. And if every fire alarm system has to have a smoke alarm and no fire alarm systems have smoke alarms and no office buildings can be equipped with a fire alarm system and have no reason to exist.

   Comment: None

5. *Problematic statement*: Anything that is mounted on something is a supplementary device.

   Correction: Remove statement 2.

   Comment: I think that part of the problem is that, in deciding how to correct the ontology, one really needs domain knowledge. There are various ways to resolve it logically—hard to know which is the correct one.

QUESTION 4.4 (PATTERN 4 EXPLANATION 4)

EXPLANATION:

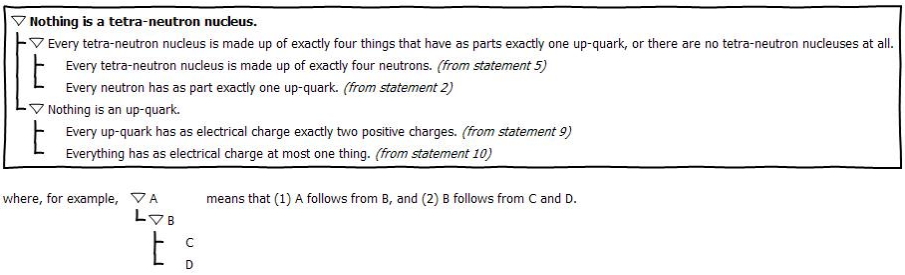


Figure B.19: The explanation in Question 4.4 (pattern 4 explanation 4), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Everything has an electrical charge.

   Comment: Two positive charges is not the way I would say it. I would say it as a positive charge of two meaning a singular charge of two, instead of referring to the charges as two separate things.

2. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Statement 10: "Everything has as electrical charge at most one thing" seems to be the least well defined in this example. It's also a limiting factor that everything has an electrical charge of one. Perhaps change it to a better number?

   Comment: I'd say the part that was the hardest was wrapping my mind around the terminology. I'm unfamiliar with all things atomic so I didn't really follow it all that well.

3. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Everything has as electrical charge one or more things.

   Comment: It used scientific language that I'm not familiar with.

4. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Statement 10 is the contradict to statement 9 (?)

   Comment: The language is not nature enough.

QUESTION 5.4 (PATTERN 5 EXPLANATION 4)

EXPLANATION:

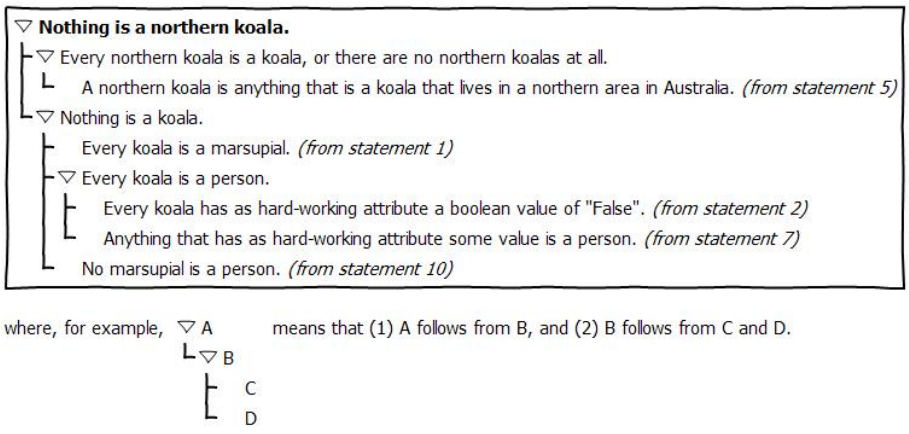


Figure B.20: The explanation in Question 5.4 (pattern 5 explanation 4), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Any value that something has as hard-working attribute is a boolean value.

   Correction: I simply think Statement 6 is out of place and adds nothing to the chain, it may not fix it.

   Comment: Statement 6 threw me and made me wonder how it played into the logical conclusion.

QUESTION 1.5 (PATTERN 1 EXPLANATION 5)

EXPLANATION:

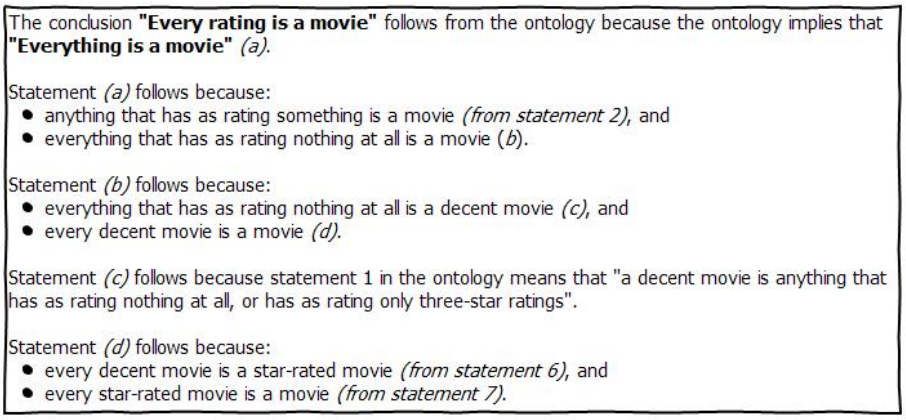


Figure B.21: The explanation in Question 1.5 (pattern 1 explanation 5), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: I would change it to “All movies have something as rating”.

   Comment: I guess all the other information besides “Everything is a movie” which seems to be shown by the first two statements, confused the issue for me.

2. *Problematic statement*: None of the statements

   Correction: None

   Comment: It says statement (c) follows because statement 1 of the ontology “means” that every movie that has a rating “nothing at all” (btw using quotes would make the rating statements more understandable) is a decent movie. Statement 1 does not directly contain this assertion in the text above. It is not clear whether it is

implied in some other way. Statement 1 says "A decent movie is anything that has as rating only three-star ratings". The meaning of "only" is not very clear to me but I would say that a movie is decent only if it has a three stars rating. I see no mention about having nothing at all as rating.

3. *Problematic statement*: Anything that has as rating something is a movie.

   Correction: This statement can be removed, because every star-rated movie statement more accurately captures this idea without potentially including actors or directors if they, or other kinds of things are allowed to be reviewed. Conversely, if the only thing that could be reviewed are movies, the statement would be accurate, if limiting to the expansion of the ontology.

   Comment: None

QUESTION 2.5 (PATTERN 2 EXPLANATION 5)

EXPLANATION:

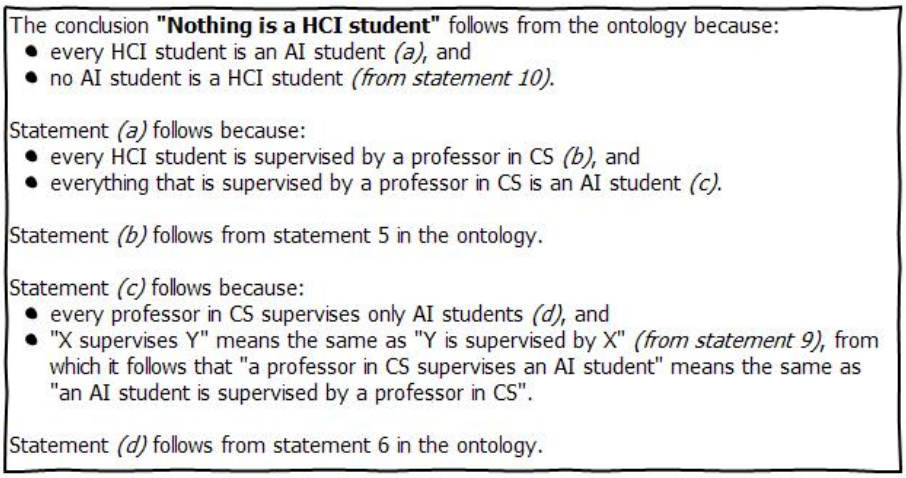


Figure B.22: The explanation in Question 2.5 (pattern 2 explanation 5), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Every professor in CS is a faculty member that supervises students.

   Comment: I think this is hard because this takes a while to understand that X supervises Y = as Y is supervised by X. You want to think "is supervised by".

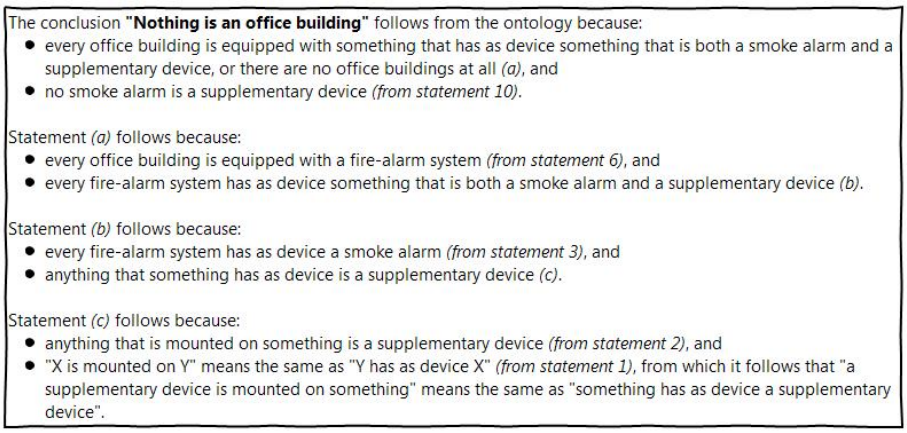
2. *Problematic statement*: Every professor in CS is a faculty member that supervises only AI students.

   Correction: Every professor in CS is a faculty member that supervises either AI or HCI students.

   Comment: The definitions were a little disorienting to me personally.

QUESTION 3.5 (PATTERN 3 EXPLANATION 5)

EXPLANATION:


The conclusion **"Nothing is an office building"** follows from the ontology because:
- every office building is equipped with something that has as device something that is both a smoke alarm and a supplementary device, or there are no office buildings at all *(a)*, and
- no smoke alarm is a supplementary device *(from statement 10)*.

Statement *(a)* follows because:
- every office building is equipped with a fire-alarm system *(from statement 6)*, and
- every fire-alarm system has as device something that is both a smoke alarm and a supplementary device *(b)*.

Statement *(b)* follows because:
- every fire-alarm system has as device a smoke alarm *(from statement 3)*, and
- anything that something has as device is a supplementary device *(c)*.

Statement *(c)* follows because:
- anything that is mounted on something is a supplementary device *(from statement 2)*, and
- "X is mounted on Y" means the same as "Y has as device X" *(from statement 1)*, from which it follows that "a supplementary device is mounted on something" means the same as "something has as device a supplementary device".

Figure B.23: The explanation in Question 3.5 (pattern 3 explanation 5), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: No smoke alarm is a supplementary device.

   Correction: It could be corrected by changing it to A smoke alarm isn't a supplementary device.

   Comment: Once I read the explanation from the bottom to the top it made far more sense and was easier to understand.

QUESTION 4.5 (PATTERN 4 EXPLANATION 5)

EXPLANATION:

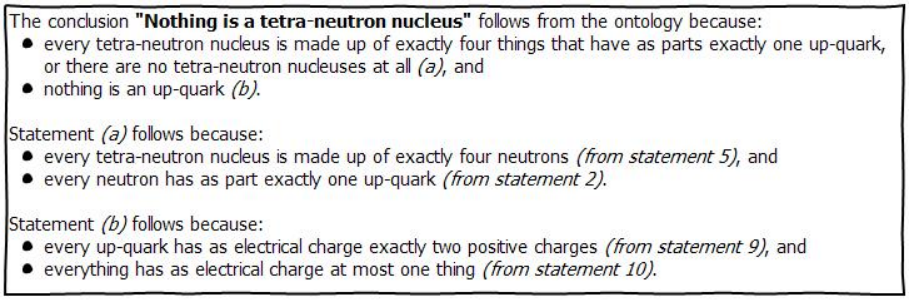


Figure B.24: The explanation in Question 4.5 (pattern 4 explanation 5), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: I think statement 10 should say that everything has an electrical charge at least one thing.

   Comment: I don't really understand what I just read!

2. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Everything has an electric charge which is equivalent to positive charges minus negative charges.

   Comment: They are somewhat confused due to typically you want to define all components not just up quarks. There are no charges assigned to down quarks so that makes the logic incomplete.

3. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction:I 'm not sure how to change it to be correct. I don't know enough about electrical charges to know if 10 is true or not. You could fix the ontology by just taking out the phrase "at most one thing" but that doesn't necessarily make it true.

   Comment: None

4. *Problematic statement*: Everything has as electrical charge at most one thing.

   Correction: Everything has one value for its electrical charge.

   Comment: A complex chain of logic is involved. Also, part of the problem is in the definition of 'has as electrical charge'. What is the range of this. Is it a set of values {-1, 1, 0, +1, +2} or is it {one positive charge; two positive charges}? An alternative might be to change statement 9 - perhaps to "one positive charge"—if one had the domain knowledge.

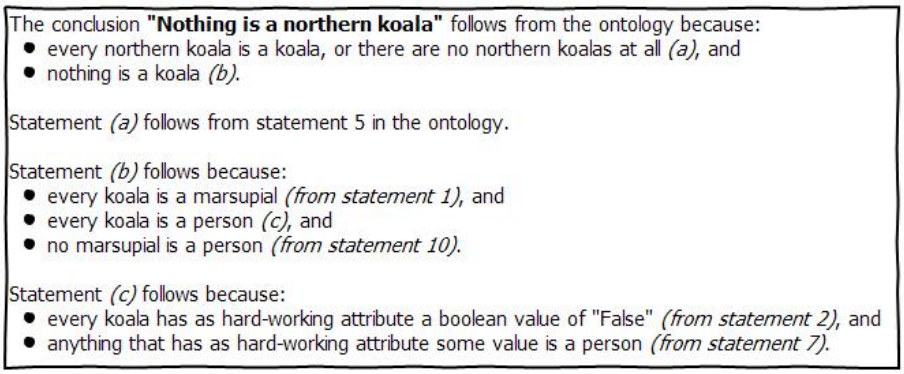
5. *Problematic statement*: Every up-quark has as electrical charge exactly two positive charges.

   Correction: By changing statement 9 to say every up-quark has as electrical charge exactly 1 positive charge, then a tetra-neutron nucleus could exist.

   Comment: This was more difficult to determine which statement should be changed because it seemed at first glance that many statements could be changed to allow the requested conclusion to be true, but after re-reading statements following my initial proposed adjustment, it appeared that the statement could still be false.

QUESTION 5.5 (PATTERN 5 EXPLANATION 5)

EXPLANATION:


The conclusion **"Nothing is a northern koala"** follows from the ontology because:
- every northern koala is a koala, or there are no northern koalas at all *(a)*, and
- nothing is a koala *(b)*.

Statement *(a)* follows from statement 5 in the ontology.

Statement *(b)* follows because:
- every koala is a marsupial *(from statement 1)*, and
- every koala is a person *(c)*, and
- no marsupial is a person *(from statement 10)*.

Statement *(c)* follows because:
- every koala has as hard-working attribute a boolean value of "False" *(from statement 2)*, and
- anything that has as hard-working attribute some value is a person *(from statement 7)*.

Figure B.25: The explanation in Question 5.5 (pattern 5 explanation 5), the indexes of axioms are from the associated justification in Table 10.1

PROBLEMATIC STATEMENTS, CORRECTIONS, AND COMMENTS:

1. *Problematic statement*: Anything that has as hard-working attribute some value is a person.

   Correction: Every person has as hard working attribute some value.

   Comment: The translation to English was a bit unusual. Specifically the statements dealing with “has as hard-working attribute some value”. This isn’t proper English grammar, so it made the exercise confusing.