Measuring the understandability of deduction rules for OWL

Conference or Workshop Item

Measuring the Understandability of Deduction Rules for OWL

Tu Anh T. Nguyen, Richard Power, Paul Piwek, Sandra Williams

Department of Computing, The Open University, UK
{t.nguyen,r.power,p.piwek,s.h.williams}@open.ac.uk

Abstract. Debugging OWL ontologies can be aided with automated reasoners that generate entailments, including undesirable ones. This information is, however, only useful if developers understand *why* the entailments hold. To support domain experts (with limited knowledge of OWL), we are developing a system that explains, in English, why an entailment follows from an ontology. In planning such explanations, our system starts from a justification of the entailment and constructs a proof tree including intermediate statements that link the justification to the entailment. Proof trees are constructed from a set of intuitively plausible deduction rules. We here report on a study in which we collected empirical frequency data on the understandability of the deduction rules, resulting in a *facility index* for each rule. This measure forms the basis for making a principled choice among alternative explanations, and identifying steps in the explanation that are likely to require extra elucidation.

Keywords: Explanations, Entailments, Justifications, Understandability, Difficulty, Deduction Rules, Inference Rules

1 Introduction

An important tool in debugging ontologies is to inspect the entailments generated by automated reasoners such as FaCT++ [13] and Pellet [12]. An obviously incorrect entailed statement such as *SubClassOf(Person,Movie)* ("Every person is a movie") signals that something has gone wrong. However, many developers, especially those with limited knowledge of OWL, will need more information in order to make the necessary corrections: they need to understand *why* this undesirable entailment follows from the ontology, before they can start to repair it. Various *axiom pinpointing* tools have been proposed to compute *justifications* of an entailment—defined as any minimal subset of the ontology from which the entailment can be drawn—including both reasoner-dependent approaches [11, 3] and reasoner-independent approaches [9, 8]. A justification provides a set of premises for an entailment, so is helpful for diagnosing an erroneous entailment; however, user studies have shown that in many cases even OWL experts are unable to work out how the conclusion follows from the premises without further explanations [7]. Moreover, the opacity of standard OWL formalisms, which are designed for efficient processing by computer programs and not for fast comprehension by people, can be another obstacle for domain experts. As a possible solution to this problem, we are developing a system that explains, in English, why an entailment follows from an ontology.

To generate such explanations, our system starts from a justification of the entailment, which can be computed using the method described by Kalyanpur et al. [9], and constructs *proof trees* in which the root node is the entailment, the terminal nodes are the axioms in the justification, and other nodes are intermediate statements (i.e., lemmas). Proof trees are constructed from a set of intuitively plausible *deduction rules* which account for a large collection of deduction patterns, with each lemma introduced by a rule (as will be described in Section 2). For a given justification, the deduction rules might allow several proof trees, in which case we need a criterion for choosing the best.[1] From the selected proof tree, the system generates an English explanation. Such an explanation should be easier to understand than one based on the justification alone, as it replaces a single complex inference step with a number of simpler steps.

As an example, Table 1 shows an explanation generated by our prototype for the (obviously absurd) entailment "Every person is a movie" based on the proof tree shown in Figure 1. The key to understanding this proof lies in the step from axiom 1 to statement (c), which is an example of an inference in need of "further elucidation"—a feature not yet implemented in our prototype.[2]

Table 1. An example explanation generated by our prototype

Input	**Entailment:** *SubClassOf(Person,Movie)* **Justification:** 1. *EquivalentClasses(GoodMovie,ObjectAllValuesFrom(hasRating,FourStars))* 2. *ObjectPropertyDomain(hasRating,Movie)* 3. *SubClassOf(GoodMovie,StarRatedMovie)* 4. *SubClassOf(StarRatedMovie,Movie)*
Output	Every person is a movie because the ontology implies that everything is a movie. Everything is a movie because (a) everything that has a rating is a movie, and (b) everything that has no rating at all is a movie. Statement (a) is from axiom 2 in the justification. Statement (b) is implied because (c) everything that has no rating at all is a good movie, and (d) every good movie is a movie. Statement (c) is implied because axiom 1 in the justification states that "a good movie is anything that has as rating only four stars". Statement (d) is implied because (e) every good movie is a star rated movie, and (f) every star rated movie is a movie. Statements (e) and (f) are from axioms 3 and 4 in the justification.

[1] Alternatively the deduction rules might not yield any proof trees, in which case the system has to fall back on simply verbalising the justification. Obviously such cases will become rarer as we expand the set of rules.

[2] See Section 4.3 for the detailed explanation of this inference step.

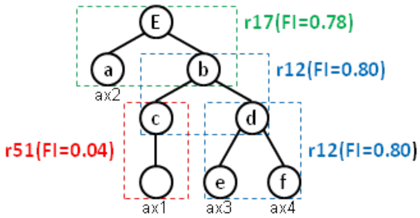

Fig. 1. The proof tree of the explanation in Table 1. The labels r17 etc. refer to rules listed in Table 3. FI values represent how easy it is to understand the rules—their *Facility Indexes*—with values ranging from 0.0 (hardest) to 1.0 (easiest).

It is important to note that there may be multiple justifications for a given entailment in an ontology, and also multiple proof trees for a given justification. For either or both of these reasons, there may be multiple potential explanations for a given entailment, some of which may be easier to follow than others. Therefore, being able to identify the most understandable proof tree among alternatives would be of great help in planning an effective explanation.

This paper focusses on the deduction rules and their understandability. We describe how our current set of deduction rules was collected through analysis of a large corpus of approximately 500 OWL ontologies, and report on an empirical study that allows us to assign easiness levels to the deduction rules.[3] These *facility indexes* provide a basis for measuring the understandability of an entire proof tree, the basis for a verbal explanation, and for making a principled choice among alternative explanations. They also indicate which steps in an explanation are likely to be difficult and in need of extra elucidation—for example, the inference from axiom 1 to statement (c) in the explanation in Table 1. We envisage that the method described here can be used by others to empirically test different sets of deduction rules.

2 Deduction Rules

Intuitively, a deduction rule is an inferential step from premises to a conclusion, which cannot be effectively simplified by introducing substeps (and hence, intermediate conclusions). This means that deduction rules should not have too many premises, and in fact we limit the number of premises to four. Formally speaking, both the conclusion and the premises are OWL axioms, but they are generalised by using variables that abstract over class and property names. An example of our deduction rules is *SubClassOf(X,Y)* $\wedge$ *SubClassOf(Y,Z)* $\rightarrow$ *SubClassOf(X,Z)* (rule 12 in Table 3), which corresponds to the well-know syllogism that from "Every X is a Y" and "Every Y is a Z" we may infer "Every X is a Z".

[3] In a deduction rule, the premises can be viewed as a justification of the entailment. Horridge et al. proposed a model for measuring the difficulty of a justification [5], but this model was based on the complexity of its logical structure of the justification rather than its difficulty for people.

Our deduction rules were derived empirically through a corpus study of around 500 OWL ontologies. These were collected from a variety of sources, including the TONES repository [2], the Swoogle search engine [4] and the Ontology Design Patterns corpus [1]; they thus cover a wide range of topics and authoring styles. To collect deduction rules, we first computed entailment-justification pairs using the method described by Nguyen et al. [10], then collated them to obtain a list of deduction patterns ranked by frequency. From this list, we selected deduction patterns that were simple (in a sense that will be explained shortly) and frequent, such as rule r12 mentioned above. Subsequently we added some further 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 algorithms.[4] An example of such rules is *ObjPropDom(r0,X)* $\wedge$ *SubClassOf(ObjectAllValuesFrom(r0,*$\perp$*),X)* $\rightarrow$ *SubClassOf(*$\top$*,X)* (rule 17), which is from "Everything that r0 something is an X" and "Everything that r0 nothing at all is an X" we infer "Everything is an X".

As a criterion of *simplicity* we considered the number of premises (we stipulated not more than four), and also what is called the *laconic* property [6]—that is, the premises in a rule should not contain information that is not required for the entailment to hold. We have assumed that deduction rules that are simple in this sense are more likely to be *understandable* by most people, as in rules 12 and 17. However, we also created non-laconic rules for unpacking part of the meaning of an axiom—the part that actually contributes to the entailment. These rules always have only one premise, but are not always obvious. An example of such rules is *EquivalentClasses(X,ObjectAllValuesFrom(r0,Y)* $\rightarrow$ *SubClassOf(ObjectAllValuesFrom(r0,*$\perp$*),X)* (rule 51).

From the deduction rules, proof trees can be constructed. As discussed before, a proof tree, as shown in Figure 1, is any tree linking the 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 justification already correspond to a rule, no further nodes (i.e., lemmas) need to be added. Otherwise, a proof tree can be sought by applying the rules, where possible, to the terminal nodes, so introducing lemmas and growing the tree bottom-up towards the root. Exhaustive search using this method may yield zero, one or multiple solutions. So far, we have obtained a total of 57 deduction rules. These rules are sufficient to generate proof trees for 48% of the justifications of subsumption entailments in the corpus (i.e., over 30,000 justifications).

3 Measuring Understandability

3.1 Materials

To measure the understandability of a rule, we devised a deduction problem in which premises of the rule were given in English, replacing class or property

[4] Reasoning services for OWL typically compute only some kinds of entailment, such as subclass and class membership statements, and ignore others.

variables by fictional 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". To control for response bias (i.e., favouring a positive, or a negative, answer to *any* question), we included questions for non-entailments and trivial entailments, which we will call *control questions* as opposed to *test questions*.

Our control questions were designed to be obvious to subjects who did the test seriously (rather than responding casually without reading the problem properly). Specifically, 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"). Problems consisted of premises followed by two questions, one a test question and one a control. For half the problems the correct answers were "Follows" and "Follows", for the other half "Follows" and "Does not Follow", with question order varied so that the test question sometimes preceded the control, and sometimes followed it.

3.2 Method

The study was conducted on CrowdFlower, a crowdsourcing service that allows customers to upload tasks to be passed to labour channel partners such as Amazon Mechanical Turk.[5] We set up the operation so that 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 respond casually without considering the problem seriously), we used CrowdFlower's quality control service which is based on *gold-standard data*: we provided problems called *gold units* for which the correct answer is specified, allowing CrowdFlower to filter automatically any subjects whose performance on gold units falls below a threshold (75%). Our gold units resembled our test units, each having premises followed by two questions, but both questions were control questions with answers that should be obvious to any serious subject. The management of gold units is internal to CrowdFlower, so these data are not included in our analysis.

Of the 57 deduction rules we collected, 51 rules were measured. For example, rule 17 (from Figure 1) was measured using data gathered from the problem in Figure 2, with the rule's entailment as the second question. 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 non-gold problems to subjects until it collects up to a specified number of valid responses for each problem. In our study we specified 50, but since some subjects gave up part-way through, the number of subjects was over 100.

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

Fig. 2. The testing problem for rule r17, which is *ObjPropDom(r0,X)* $\wedge$ *SubClassOf(ObjectAllValuesFrom(r0,$\perp$),X)* $\rightarrow$ *SubClassOf($\top$,X)*

4 Results

The main aim of the study was to collect frequency data on whether people recognise that the conclusion of a (verbalised) deduction rule follows from the premises. However, these data provide a valid index of understandability only if we can control for positive response bias: in the extreme case, a subject that always gives the positive answer ("Follows" rather than "Does not follow") will get all the test questions right, regardless of their difficulty. We used control questions to address this issue—additional to the CrowdFlower gold-unit filtering. The use of control questions in each problem also provided an opportunity for confirming that in general subjects were taking the survey seriously.

4.1 Control Questions

Figure 3 shows that for the 108 subjects that participated in the study, all answered around 75% or more of the control questions correctly, suggesting that they were performing the task seriously.

4.2 Response Bias

Table 2 shows the absolute frequencies of the responses "Follows" (+F) and "Does not follow" (−F) for all non-gold 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 for half the problems the correct answers were +F+F, while for half they were +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

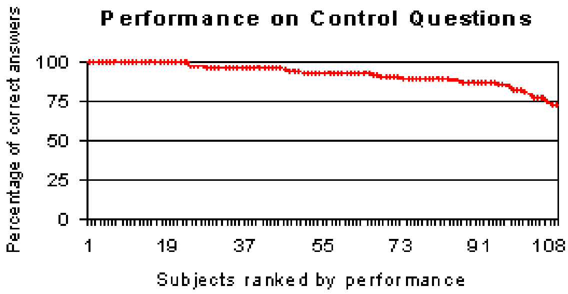

Fig. 3. The subjects' performance on the control questions

would expect an overall rate higher than this, but in fact we obtain 2823/4930 or 57.3%, suggesting little or no bias in either direction.

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 (χ^2 = 1116.3, df = 1, p < 0.0001).

Table 2. The distribution of the subjects' performance

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

4.3 Facility Indexes

We use the proportion of correct answers for each test question as an index of understandability of the associated deduction rule, which we call its *facility index*. This index provides our best estimate of the probability that a person will understand the relevant inference step—i.e., that they will recognise that the conclusion follows from the premises—and accordingly ranges from 0.0 to 1.0. Values of the facility index for the rules tested in the study are shown in Table 3, ordered from high values to low. In this table, the rules r12 and r17 used in the explanation in Table 1 are relatively easy, with facility indexes of 0.80 and 0.78. By contrast rule r51, which infers statement (c) from axiom 1 in the example, is the hardest, with a facility index of only 0.04, and hence evidently a step in need of further elucidation—for instance as follows:

> Statement (c) is inferred from axiom 1, which asserts an equivalence between two classes: 'good movie' and 'anything that has as rating only four stars'. Since the second class trivially accepts anything that has no rating at all, we conclude that anything that has no rating at all is a good movie.

It can also be seen in Table 3 that for closely related rules, such as r11, r12 and r14, the facility indexes are quite close to each other (see also r20 and r21, r45 and r46, r49 and r50), a result that confirms the reliability of the values. In general, the subjects find it difficult to understand inference steps that conclude a concept is unsatisfiable (as in rules 50, 49...) and those related to the unfamiliar behaviour of the universal quantifier in OWL (as in rule 51).

5 Conclusion

The main aim of this paper is to report empirical results on the understandability of some inference steps that often occur in proofs, in particular for entailments computed from ontologies. These results allow us to estimate the understandability of proofs that can serve as the basis for verbal explanations of entailments, so making it clear to an ontology developer why an undesired statement was inferred, and which axiom(s) in the ontology should be removed or revised.

In our explanation planner, the facility indexes for the deduction rules are used in two ways. First, by combining the values for all the rules in a given proof tree, we can estimate the difficulty of the whole tree, and thus make a principled choice among alternative trees. If we think of facility indexes as measuring the probability that a reader will understand a given step in the explanation, a natural method of combining indexes would be to multiply them, so computing the joint probability of all steps being followed; the higher this value, obviously, the better. Second, once a proof tree has been chosen as more understandable than the alternatives (if any), we can apply the indexes again by looking for steps that are relatively hard, and considering whether to add extra elucidation at that point. We plan to do this by investigating a range of explanation strategies for each difficult rule, and determining empirically which is most effective.

Leaving aside the way facility indexes are used in our work, we believe both the indexes and our method for obtaining them are worth reporting as a resource for other researchers, who might find them useful in alternative models or contexts.

References

1. Ontology Design Patterns. `http://ontologydesignpatterns.org`, Last Accessed: 30th August 2010
2. The TONES Ontology Repository. `http://owl.cs.manchester.ac.uk/repository/`, Last Accessed: 30th August 2010

3. Baader, F., Hollunder, B.: Embedding Defaults into Terminological Knowledge Representation Formalisms. Journal of Automated Reasoning 14, 149–180 (1995)
4. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: ACM International Conference on Information and Knowledge Management (CIKM 2004). pp. 652–659 (2004)
5. Horridge, M., Bail, S., Parsia, B., Sattler, U.: The Cognitive Complexity of OWL Justifications. In: International Semantic Web Conference (ISWC 2011). pp. 241–256 (2011)
6. Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: International Semantic Web Conference (ISWC 2008). pp. 323–338 (2008)
7. Horridge, M., Parsia, B., Sattler, U.: Lemmas for Justifications in OWL. In: International Workshop on Description Logics (DL 2009) (2009)
8. Ji, Q., Qi, G., Haase, P.: A Relevance-Directed Algorithm for Finding Justifications of DL Entailments. In: ASWC 2009, Asian Semantic Web Conference. pp. 306–320. Berlin, Heidelberg (2009)
9. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding All Justifications of OWL DL Entailments. In: International Semantic Web Conference (ISWC 2007) (2007)
10. Nguyen, T.A.T., Piwek, P., Power, R., Williams, S.: Justification Patterns for OWL DL Ontologies. Tech. Rep. TR2011/05, The Open University, UK (2010)
11. Schlobach, S., Cornet, R.: Non-standard Reasoning Services for the Debugging of Description Logic Terminologies. In: International Joint Conference on Artificial Intelligence (IJCAI 2003). pp. 355–360 (2003)
12. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics 5, 51–53 (2007)
13. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: International Joint Conference on Automated Reasoning (IJCAR 2006). pp. 292–297 (2006)

Table 3: 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 short, the names of OWL functors are abbreviated.

ID	Rule	Deduction Problem	CA	S	FI
1	EqvCla(X,Y...) →SubClaOf(X,Y)	A hiatea is defined as a milvorn. →Every hiatea is a milvorn.	49	49	1.00
2	SubClaOf(X,ObjIntOf(Y,Z...)) →SubClaOf(X,Y)	Every ormyrr is both a gargoyle and a harpy. →Every ormyrr is a gargoyle.	47	49	0.96
3	ObjPropDom(r0,X) ∧ SubClaOf(X,Y) →ObjPropDom(r0,Y)	Anything that has a supernatural ability is a bulette. Every bulette is a manticore. →Anything that has a supernatural ability is a manticore.	45	47	0.96
4	SubClaOf(ObjUniOf(X,Y...),Z) →SubClaOf(X,Z)	Everything that is a volodni or a treant is a maradan. →Every volodni is a maradan.	44	46	0.96
5	SubClaOf(X,Y) ∧ SubClaOf(X,Z) →SubClaOf(X,ObjIntOf(Y,Z))	Every bullywug is a grippli. Every bullywug is a prismatic. →Every bullywug is both a grippli and a prismatic.	45	48	0.94
6	SubClaOf(⊤,Y) →SubClaOf(X,Y)	Everything is a kelpie. →Every person is a kelpie.	43	46	0.93
7	SubClaOf(X,ObjSomValF(r0,⊤)) ∧ SubClaOf(X,ObjAllValF(r0,Y)) →SubClaOf(X,ObjSomValF(r0,Y))	Every locathah eats something. Every locathah eats only orogs. →Every locathah eats an orog.	45	50	0.90
8	ObjPropRng(r0,X) ∧ SubClaOf(X,Y) →ObjPropRng(r0,Y)	Anything that something lives in is a tarrasque. Every tarrasque is a kraken. →Anything that something lives in is a kraken.	44	49	0.90
9	ObjPropDom(r0,Y) ∧ SubClaOf(X,ObjSomValF(r0,Z)) →SubClaOf(X,Y)	Anything that is a messenger of something is a landwyrm. Every spellgaunt is a messenger of a gravorg. →Every spellgaunt is a landwyrm.	43	50	0.86
10	EqvCla(X,ObjUniOf(Y,Z...)) →SubClaOf(Y,X)	Every cooshee is a peryton or a banderlog; everything that is a peryton or a banderlog is a cooshee. →Every peryton is a cooshee.	41	50	0.82
11	SubClaOf(X,ObjSomValF(r0,Y)) ∧ SubClaOf(ObjMinCard(1,r0,Y),Z)) →SubClaOf(X,Z)	Every varag lives on a seaplane. Everything that lives on at least one seaplane is an urophion. →Every varag is an urophion.	40	49	0.82
12	SubClaOf(X,Y) ∧ SubClaOf(Y,Z) →SubClaOf(X,Z)	Every sivak is a draconian. Every draconian is a guulvorg. →Every sivak is a guulvorg.	37	46	0.80
13	SubClaOf(X,ObjCompOf(X)) →SubClaOf(X,⊥)	Every zezir is something that is not a zezir. →Nothing is a zezir.	40	50	0.80
14	SubObjPpOf(r0,r1) ∧ SubObjPpOf(r1,r2) →SubObjPpOf(r0,r2)	The property "is a kobold of" is a sub-property of "is a drow of". The property "is a drow of" is a sub-property of "is a tiefling of". →The property "is a kobold of" is a sub-property of "is a tiefling of".	41	52	0.79
15	SubClaOf(X,ObjSomValF(r0,Y)) ∧ SubClaOf(Y,Z) →SubClaOf(X,ObjSomValF(r0,Z))	Every phaerlin is father of a firbolg. Every firbolg is a gnoll. →Every phaerlin is father of a gnoll.	37	47	0.79
16	EqvCla(X,ObjIntOf(Y,Z...)) →SubClaOf(X,Y)	A cyclops is anything that is both a troofer and a gathra. →Every cyclops is a troofer.	37	47	0.79
17	ObjPropDom(r0,X) ∧ SubClaOf(ObjAllValF(r0,⊥),X) →SubClaOf(⊤,X)	Everything that has a worship leader is a fomorian. Everything that has no worship leader at all is a fomorian. →Everything is a fomorian.	39	50	0.78
18	ObjPropRng(r0,X) ∧ SymObjProp(r0) →ObjPropDom(r0,X)	Anything that something is an abrian of is a grolantor. X is an abrian of Y if and only if Y is an abrian of X. →Anything that is a sibling of something is a grolantor.	36	47	0.77
19	SubClaOf(Y,X) ∧ SubClaOf(ObjCompOf(Y),X) →SubClaOf(⊤,X)	Every oblivion moss is a vegepygmy. Everything that is not an oblivion moss is a vegepygmy. →Everything is a vegepygmy.	36	47	0.77
20	ObjPropDom(r0,⊥)	There does not exist anything that is a grimlock of something.	39	51	0.76

Continued on Next Page...

	→SubClaOf(⊤,ObjAllValF(r0,⊥))	→Everything is not a grimlock.			
21	ObjPropRng(r0,⊥) →SubClaOf(⊤,ObjAllValF(r0,⊥))	There does not exist anything that something has as a catter. →Everything has no catter at all.	37	49	0.76
22	DisCla(X,Y...) ∧ SubClaOf(Z,X) ∧ SubClaOf(W,Y) →DisCla(Z,W)	No plant is an animal. Every kalamanthis is a plant. Every tendriculos is an animal. →No kalamanthis is a tendriculos.	35	46	0.76
23	SubClaOf(X,ObjSomValF(r0,Y)) ∧ SubClaOf(Y,ObjSomValF(r0,Z)) ∧ TrnObjProp(r0) →SubClaOf(X,ObjSomValF(r0,Z))	Every dero is a tendriculos of a harpy. Every harpy is a tendriculos of a tasloi. If X is a tendriculos of Y and Y is a tendriculos of Z then X is a tendriculos of Z. →Every dero is a tendriculos of a tasloi.	38	51	0.75
24	SubClaOf(X,ObjUniOf(Y,Z)) ∧ SubClaOf(Y,W) ∧ SubClaOf(Z,W) →SubClaOf(X,W)	Every mongrelfolk is a nilbog or a norker. Every nilbog is a skulk. Every norker is a skulk. →Every mongrelfolk is a skulk.	35	48	0.73
25	SubClaOf(ObjCompOf(X),Y) →SubClaOf(⊤,ObjUniOf(X,Y))	Everything that is not a spriggan is an orog. →Everything is a spriggan or an orog.	36	50	0.72
26	SubClaOf(X,ObjUniOf(Y,Z)) ∧ SubClaOf(Y,Z) →SubClaOf(X,Z)	Every merfolk is a lizardfolk or a kobold. Every lizardfolk is a kobold. →Every merfolk is a kobold.	35	49	0.71
27	SubClaOf(ObjSomValF(r0,X),Y) ∧ SubClaOf(ObjAllValF(r0,⊥),Y) →SubClaOf(ObjAllValF(r0,X),Y)	Everything that supervises a worg is a stirge. Everything that supervises nothing at all is a stirge. →Everything that supervises only worgs is a stirge.	35	49	0.71
28	ObjPropDom(r0,X) ∧ SymObjProp(r0) →ObjPropRng(r0,X)	Anything that is an obliviax of something is a kraken. X is an obliviax of Y if and only if Y is an obliviax of X. →Anything that something is an obliviax of is a kraken.	34	49	0.69
29	SubClaOf(X,ObjSomValF(r0,ObjSomValF(r0,Y))) ∧ TrnObjProp(r0) →SubClaOf(X,ObjSomValF(r0,Y))	Every draconian is a spriggan of something that is a spriggan of a shifter. If X is a spriggan of Y and Y is a spriggan of Z then X is a spriggan of Z. →Every draconian is a spriggan of a shifter.	34	50	0.68
30	ObjPropRng(r0,Z) ∧ SubClaOf(X,ObjSomValF(r0,Y)) →SubClaOf(X,ObjSomValF(r0,ObjIntOf(Y,Z)))	Anything that something resembles is a corollax. Every mudmaw resembles a jermlaine. →Every mudmaw resembles something that is both a jermlaine and a corollax	32	50	0.64
31	SubClaOf(⊤,Y) ∧ DisCla(X,Y) →SubClaOf(X,⊥)	Everything is a darfellan. No grippli is a darfellan. →Nothing is a grippli.	30	47	0.64
32	SubClaOf(X,ObjExtCard(n1,r0,Y)) →SubClaOf(X,ObjMinCard(n2,r0,Y)), where n2≤n1	Every oaken defender has exactly two dry leaves. →Every oaken defender has at least one dry leaf.	29	46	0.63
33	ObjPropDom(r0,X) ∧ SubObjPpOf(r1,r0) →ObjPropDom(r1,X)	Anything that gyres something is a tiefling. The property "raths" is a sub-property of "gyres". →Anything that raths something is a tiefling.	28	46	0.61
34	SubClaOf(X,Y) ∧ DisCla(X,Y) →SubClaOf(X,⊥)	Every aasimar is a sirine. No aasimar is a sirine. →Nothing is an aasimar.	30	53	0.57
35	SubClaOf(X,Y) ∧ SubClaOf(X,Z) ∧ DisCla(Y,Z) →SubClaOf(X,⊥)	Every needleman is a basidirond. Every needleman is a battlebriar. No basidirond is a battlebriar. →Nothing is a needleman.	27	48	0.56
36	TrnObjProp(r0) ∧ InvObjProp(r0,r1) →TrnObjProp(r1)	If X toves Y and Y toves Z then X toves Z. X toves Y if and only if Y is toved by X. →If X is toved by Y and Y is toved by Z then X is toved by Z.	27	49	0.55
37	SubClaOf(X,ObjSomValF(r0,Y)) ∧ SubObjPropOf(r0,r1) →SubClaOf(X,ObjSomValF(r1,Y))	Every halfling is an ascomoid of a kenku. The property "is an ascomoid of" is a sub-property of "is a basidirond of". →Every halfling is a basidirond of a kenku.	28	51	0.55
38	ObjPropRng(r0,X)	Anything that something brilligs is a girallon.	24	46	0.52

Continued on Next Page...

	∧ SubObjPropOf(r1,r0) →ObjPropRng(r1,X)	The property "gimbles" is a sub-property of "brilligs". →Anything that something gimbles is a girallon.			
39	SubClaOf(X,Y) ∧ SubClaOf(X,ObjCompOf(Y)) →SubClaOf(X,⊥)	Every darkmantle is a gorgon. Every darkmantle is not a gorgon. →Nothing is a darkmantle.	25	49	0.51
40	SubClaOf(X,ObjSomValF(r0,ObjIntOf(Y,Z...))) ∧ DisCla(Y,Z) →SubClaOf(X,⊥)	Every daemonfey is preceded by something that is both an axani and a phoera. No axani is a phoera. →Nothing is a daemonfey.	25	50	0.50
41	SubClaOf(X,ObjMinCard(n1,r0,Dor⊤)) ∧ SubClaOf(X,ObjMinCard(n2,r0,⊤)), where n2<n1 →SubClaOf(X,⊥)	Every jermlaine possesses at least three things. Every jermlaine possesses at most one thing. →Nothing is a jermlaine.	22	46	0.48
42	SubClaOf(X,ObjSomValF(r0,Y)) ∧ SubClaOf(Y,⊥) →SubClaOf(X,⊥)	Every tasloi has as owner an aasimar. Nothing is an aasimar. →Nothing is a tasloi.	20	44	0.45
43	FunDataProp(d0) ∧ SubClaOf(X,DataMinCard(n,d0,DR0)), where n>1 →SubClaOf(X,⊥)	Everything has as ratings at most one value. Every buckawn has as ratings at least four integer values. →Nothing is a buckawn.	20	49	0.41
44	ObjPropRng(r0,X) ∧ InvObjProp(r0,r1) →ObjPropDom(r1,X)	Anything that something gimbles from is a terlen. X gimbles from Y if and only if Y gimbles into X. →Anything that gimbles into something is a terlen.	19	47	0.40
45	FunDataProp(d0) ∧ SubClaOf(X,DataHasVal(d0,l0⋆DT0)) ∧ SubClaOf(X,DataHasVal(d0,l1⋆DT1)) where DT0 and DT1 are disjoint or l0 ≠ l1 →SubClaOf(X,⊥)	Everything has as power level at most one value. Every sirine has as power level an integer value of 5. Every sirine has as power level an integer value of 7. →Nothing is a sirine.	18	45	0.40
46	FuncObjProp(r0) ∧ SubClaOf(X,ObjHasVal(r0,i0)) ∧ SubClaOf(X,ObjHasVal(r0,i1)) ∧ DiffInd(i0,i1...) →SubClaOf(X,⊥)	Everything worships at most one thing. Every selkie worships Ashur. Every selkie worships Enki. Ashur and Enki are different individuals. →Nothing is a selkie.	17	44	0.39
47	ObjPropDom(r0,X) ∧ InvObjProp(r0,r1) →ObjPropRng(r1,X)	Anything that gimbles from something is an atomie. X gimbles from Y if and only if Y gimbles into X. →Anything that something gimbles into is an atomie.	18	48	0.38
48	SubClaOf(X,ObjAllValF(r0,Y)) ∧ InvObjProp(r0,r1) →SubClaOf(ObjSomValF(r1,X),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	DataPropRng(d0,DR0) ∧ X⊑ObjSomValF(r0,DataHasVal(d0,l0⋆DT1)) where DR0 & DT1 are disjoint →SubClaOf(X,⊥)	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	DataPropRng(d0,DR0) ∧ SubClaOf(X,DataSomeValFrm(d0,DT1)) where DR0 & DT1 are disjoint →SubClaOf(X,⊥)	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	EqvCla(X,ObjAllValF(r0,Y)) →SubClaOf(ObjAllValF(r0,⊥),X)	A hiatea is anything that eats only lamias. →Everything that eats nothing at all is a hiatea.	2	49	0.04