Explaining by Example: Model Exploration for Ontology Comprehension

Johannes Bauer, Ulrike Sattler, and Bijan Parsia

School of Computer Science
The University of Manchester
Oxford Road
Manchester
M13 9PL

Abstract. In this paper, we describe an approach for ontology comprehension support called model exploration, in which models for ontologies are generated and presented interactively. We report on a pilot user study we conducted on our prototype implementation, SuperModel, to evaluate the effectiveness of model exploration in supporting users in understanding an ontology. We also discuss the issues involved in using tableau reasoners for the generation of models for model exploration.

1 Motivation

At different stages of an ontology's life-cycle, the people working with it require some sort of understanding of it; e.g. which parts of it encode what piece of knowledge explicitly or implicitly, how it is meant to be used on its own or in conjunction with other ontologies, and if and where it is broken and how it may be fixed. Tool support is necessary to achieve these kinds of understanding, as ontologies can be as broad and complicated as the fields of knowledge they model, and as it is easy to get lost in the highly complex interactions between the large numbers of logical axioms they comprise.

This need is testified and partially met by the impressive tool support which has recently become available. However, support facilitates use and use spawns demand for more support. Thus, the very increase in popularity of ontology engineering brought about by ontology editors, visualization, justifications, etc. may be seen as one reason for the demand for even more support.

Looking at anecdotal evidence, we found that some DL experts liked to use pen and paper to visualize models of ontologies to explain or check some of their lesser obvious implications. Our idea was that the automatic generation and presentation of such models could similarly aid the understanding of ontologies.

Think, for example, of an ontology stating that a `Nobleman` is someone who is the son of a `Nobleman`, and a `Commoner` is someone who is not a `Nobleman`. One might be surprised to see that a `Commoner` can have a son who is a `Nobleman`. However, the example depicted in Fig. 1, where that `Nobleman` is at the same time the son of both a `Commoner` and a `Nobleman`, should immediately clear up

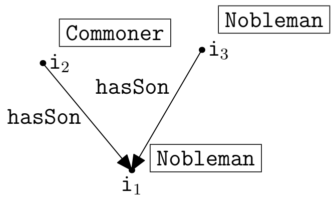

Fig. 1: A Clarifying Example

the situation and make it obvious that the ontology lacks a statement saying that no-one can be son of two men.

This paper is based on [1], in which we developed the notion of model exploration from these observations. Informally, model exploration is the activity of interactively generating and visualizing models for concepts in ontologies in order to learn something about the concept/ontology. The details are fleshed out in Section 3. In Section 6 we report on the pilot user study we conducted on our model exploration prototype, SuperModel, to get a first idea of the merits of model exploration.

2 Notation

In this paper, we use standard DL terminology, syntax and semantics as defined e.g. in [2]. Additionally, we will use $\mathtt{C}$ and $\mathtt{D}$ for concept descriptions, $\mathtt{i}$, $\mathtt{i}_1$, $\mathtt{i}_2\ldots$ for individual names, $\mathtt{r}$ for role names, $\mathtt{R}$ for roles, and $\mathcal{C}$, $\mathcal{R}$, and N_I for sets of concepts, roles, and individual names, respectively.

3 Model Exploration

The idea behind model exploration is that it may be useful for users to see example instantiations—models—of the concepts in an ontology to understand their semantics. Unfortunately, there may be very many (even infinitely many) very big (infinitely big) models of a given concept. What we hope is that seeing *small parts* of *just a few* models suffices in many cases to give users the information they seek. This is why model exploration includes an interactive component which lets the user decide which parts of what models to see.

Model exploration is an activity cycling through the stages of *model specification*, *model generation*, and *model inspection* (see Fig. 2). In the first stage, the users add axioms to an ABox which serve as constraints for the type of model they want to see next. In model generation, a completion graph for this *constraint ABox* is computed and adorned with additional information. In model inspection, a connected subgraph, the *model excerpt*, of this completion graph is presented graphically to the user and may be expanded (explored) along the edges of the completion graph.

The first constraint ABox is obtained from the user-specified *root concept* $\mathtt{C}_r$ by adding the assertion $\mathtt{C}_r(\mathtt{i}_r)$ to an empty ABox. The model excerpt always includes the *root individual* $\mathtt{i}_r$.

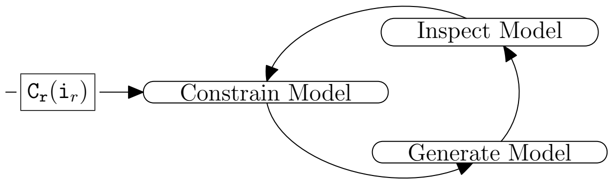

Fig. 2: Model Exploration Cycle

3.1 Status of Information

To make the presentation of the rest of this section easier, we will assume, without loss of generality, models in which, for every individual in its domain, there is an individual name which it maps to that individual. We can then refer to individuals in an interpretation by their individual name and describe interpretations and their corresponding graphs by sets of ABox axioms in a straightforward manner: we say that A *describes* an interpretation $\mathcal{I}$, if

$$\begin{aligned} A = \{&\mathtt{C}(\mathtt{i}) \mid \mathtt{C} \in \mathcal{C}, \mathtt{i} \in N_I, \mathtt{i}^{\mathcal{I}} \in \mathtt{C}^{\mathcal{I}}\} \\ &\cup \{\mathtt{R}(\mathtt{i},\mathtt{i}') \mid \mathtt{r} \in \mathcal{R}, \mathtt{i},\mathtt{i}' \in N_I, (\mathtt{i}^{\mathcal{I}}, \mathtt{i}'^{\mathcal{I}}) \in \mathtt{R}^{\mathcal{I}}\} \end{aligned}$$

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. We then define the *graph corresponding to* $\mathcal{I}$ as the graph $G_{\mathcal{I}} = (V, E, \mathcal{L}_V, \mathcal{L}_E)$ where

$$\begin{aligned} V &= \Delta^{\mathcal{I}}, & \mathcal{L}_V &: V \to \mathcal{C}^2,\ \mathcal{L}_V(\mathtt{i}) = \{\mathtt{C} \in \mathcal{C} \mid \mathtt{i} \in \mathtt{C}^{\mathcal{I}}\} \\ E &= V^2, & \mathcal{L}_E &: E \to \mathcal{R}^2,\ \mathcal{L}_E(e) = \{\mathtt{R} \in \mathcal{R} \mid e \in \mathtt{R}^{\mathcal{I}}\}. \end{aligned}$$

It is obvious that an interpretation $\mathcal{I}$ and its corresponding graph $G_{\mathcal{I}}$ are inter-translatable.

A model for an ABox must satisfy all of the ABox's axioms, and thus it is easy to see that an ABox always is a subset of every set A describing a model for it. This lets us define a function $\mathsf{asrt}()$, for 'asserted', on edge- and node labels of a graph $G_{\mathcal{I}}$ corresponding to a model $\mathcal{I}$ for an ABox $\mathcal{A}$:

Let $\mathtt{C}$ be a concept, $\mathtt{R}$ a role and $\mathtt{i}$, $\mathtt{i}'$ individual names. Then

$$\mathsf{asrt}(\mathtt{C}, \mathtt{i}^{\mathcal{I}}) = \begin{cases} 1 & \text{if } \mathtt{C}(\mathtt{i}) \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases} \quad \text{and} \quad \mathsf{asrt}(\mathtt{R}, (\mathtt{i}^{\mathcal{I}}, \mathtt{i}'^{\mathcal{I}})) = \begin{cases} 1 & \text{if } \mathtt{R}(\mathtt{i},\mathtt{i}') \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases}$$

Another observation is that, for some ABoxes, there are assertions which are in every set describing a model for these ABoxes, but not in the ABoxes themselves. In fact, we would like to generalize this: consider the models for the ABox $\mathcal{A} = \{(\exists \mathtt{r}.\mathtt{C})(\mathtt{i_r})\}$. In all models for $\mathcal{A}$, $\mathtt{i_r}$ will have an $\mathtt{r}$-successor which is a $\mathtt{C}$. Thus, all models of $\mathcal{A}$ satisfy an axiom $\mathtt{r}(\mathtt{i_r}, \mathtt{i_1})$ for varying individual names $\mathtt{i_1}$.

This is a commonality which we believe is worth pointing out when visualizing a model. Thus, we would like to define a function $\mathsf{mnd}()$, for 'mandatory', on node and edge labels such that, if, in a model, $\mathtt{i_1}$ is the name of that $\mathtt{r}$-successor, then $\mathsf{mnd}(\mathtt{r}, (\mathtt{i}_r{}^{\mathcal{I}}, \mathtt{i_1}{}^{\mathcal{I}})) = 1$ and $\mathsf{mnd}(\mathtt{C}, \mathtt{i_1}) = 1$.

Now, while asserted node and edge labels are asserted independently of each other, they can only be defined to be mandatory in relation to each other: consider the concept definition

$$\mathtt{D} \sqsubseteq \exists \mathtt{p}.\mathtt{A} \sqcap \exists \mathtt{p}.\mathtt{B} \sqcap \exists \mathtt{p}.\mathtt{C} \sqcap \leq 2\,\mathtt{p}$$

and the models depicted in Fig. 3, which are the alternatives created for $\mathtt{D}$ by a standard tableau reasoner.

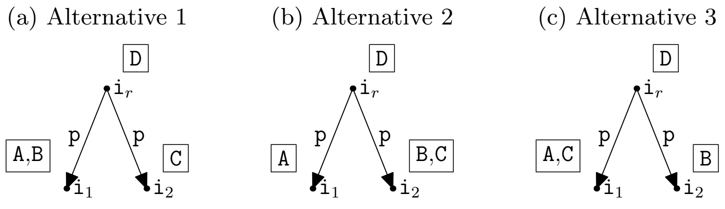

Fig. 3: Illustration of Mandatory Information

Either of the sets $\{\mathtt{D}(\mathtt{i}_r), \mathtt{p}(\mathtt{i}_r, \mathtt{i}_1), \mathtt{A}(\mathtt{i}_1)\}$ and $\{\mathtt{D}(\mathtt{i}_r), \mathtt{p}(\mathtt{i}_r, \mathtt{i}_1), \mathtt{B}(\mathtt{i}_1)\}$ could be said to be a set of mandatory axioms[1] in the sense that a similar set of axioms is satisfied by every model of $\mathtt{D}$.

No such set should contain the axioms $\mathtt{A}(\mathtt{i}_1)$ and $\mathtt{B}(\mathtt{i}_1)$, because in other models, the $\mathtt{p}$-successors of the instance of $\mathtt{D}$ which are an $\mathtt{A}$ and a $\mathtt{B}$, respectively, are not the same, as can be seen in Figs. 3b and 3c.

In order to be useful for users of model exploration, a definition for labels being mandatory needs to be simple enough to work with in practice, yet capture a meaningful notion of commonalities of all models of an ontology and a concept. We believe that the following definition strikes a good compromise:

A set of ABox axioms S is a *mandatory superset* of an ABox $\mathcal{A}$ w.r.t. an ontology $\mathcal{O}$, if S includes $\mathcal{A}$ and every model for $\mathcal{A}$ and $\mathcal{O}$ can be transformed into a model for $\mathcal{O}$ and all the axioms in S only by changing its interpretation of individual names occurring in S but not in $\mathcal{A}$ or $\mathcal{O}$.

[1] modulo renaming of individuals not occurring in ontology or constraint ABox, see [1]

For a model $\mathcal{I}$ of an ABox $\mathcal{A}$ and a mandatory superset S of $\mathcal{A}$, we define a function $\mathsf{mnd}()$ on node and edge labels $\mathtt{C} \in \mathcal{C}$ and $\mathtt{R} \in \mathcal{R}$, resp., and nodes $\{\mathtt{i}^{\mathcal{I}}, \mathtt{i}'^{\mathcal{I}}\} \subseteq \Delta^{\mathcal{I}}, \{\mathtt{i}, \mathtt{i}'\} \subseteq N_I$ in a graph corresponding to $\mathcal{I}$ as follows:

$$\mathsf{mnd}(\mathtt{C}, \mathtt{i}^{\mathcal{I}}) = \begin{cases} 1 & \text{if } \mathtt{C}(\mathtt{i}) \in S, \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \mathsf{mnd}(\mathtt{R}, (\mathtt{i}^{\mathcal{I}}, \mathtt{i}'^{\mathcal{I}})) = \begin{cases} 1 & \text{if } \mathtt{R}(\mathtt{i}, \mathtt{i}') \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Let $G_{\mathcal{I}}$ be the graph corresponding to an interpretation based on a completion graph. If that completion graph was generated by a tableau reasoner using dependency sets for backjumping [3, 4], then a function $\mathsf{mnd}()$ can be derived from information about the dependency sets of labels in the completion graph [1]: for a label of a node or edge in $G_{\mathcal{I}}$, $\mathsf{mnd}()$ is 1 exactly if the dependency set for the corresponding label in the completion graph is empty.

Note that mandatory supersets generated this way are not necessarily maximal. Consider the concept definition

$$\mathtt{D} \sqsubseteq (\mathtt{C} \sqcap \mathtt{C}') \sqcup (\mathtt{C} \sqcap \mathtt{C}'')$$

The root individual of the completion graph for $\mathtt{D}$ can be labeled $\{\mathtt{D}, \mathtt{C}, \mathtt{C}'\}$ or $\{\mathtt{D}, \mathtt{C}, \mathtt{C}''\}$, however, $\mathtt{C}$ would have been added after a non-deterministic decision of the reasoner and thus, its dependency set would be non-empty, unnecessarily excluding the axiom $\mathtt{C}(\mathtt{i}_r)$ from our mandatory superset.

Ways to extend mandatory supersets to include more axioms are left for future work.

Our prototype implementation of model exploration, SuperModel,[2] follows the approach described in this section and uses dependency sets to obtain a mandatory superset and include it in its visualization of models.

3.2 A Barbarian Example

In the user study described below, we used a simple toy ontology dealing with character generation in a rogue-like computer game. It comprises the two roles `canWield` and `hasSkillLevel` and the asserted concept hierarchy depicted in Fig. 4.

Now, suppose one were interested in knowing what `SkillLevel` a `Barbarian` needs to have to be able to wield a `Morningstar`. It would be straightforward, since one would be interested in `Barbarians`, to start exploring a model of this concept. One would thus start with a constraint ABox $\mathcal{A} = \{\mathtt{Barbarian}(\mathtt{i}_r)\}$. The model could contain two individuals, $\mathtt{i_r}$ and its `hasSkillLevel`-successor $\mathtt{i_1}$, $\mathtt{i_r}$ being a `Barbarian` and a `Role`, and $\mathtt{i_1}$ being a `SkillLevel` and, say, a `Basic`.

In the next step, the user might add the assertions $\mathtt{canWield}(\mathtt{i_r}, \mathtt{i_1})$ and $\mathtt{Morningstar}(\mathtt{i_1})$ to the constraint ABox. Fig. 5 illustrates what could be shown

[2] downloadable at http://www.man.ac.uk/∼bauerj/supermodel/

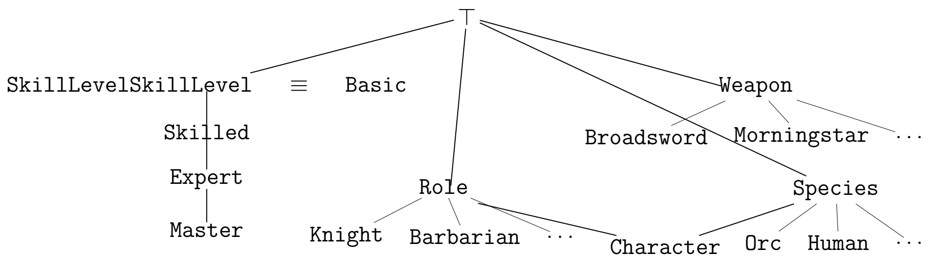

Fig. 4: Barbarian Ontology

in a model exploration system. From the information about the mandatory superset (the exclamation marks), it can be seen that, whenever something is a `Barbarian` and can wield a `Morningstar`, it must have a `SkillLevel` which is at the same time `Basic` and `Skilled`.

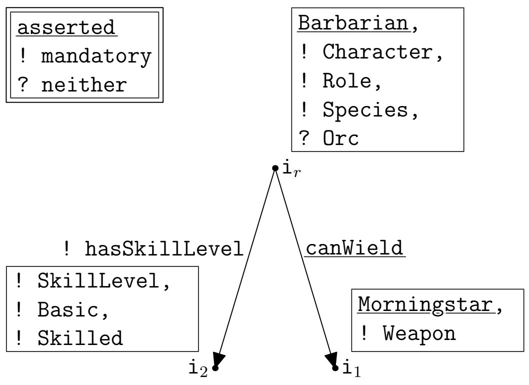

Fig. 5: Barbarian Example: Step 2

4 Model Generation

For a model to be explored, it must be generated. Since model generation is the basis of tableau reasoning in DLs, there is a large body of work dealing with exactly this problem—we used FaCT++ [5], a state-of-the-art, open-source tableau reasoner for the $\mathcal{SROIQ}(\mathbf{D})$ description logic, to generate models for model exploration.

We said model generation was *the basis* of tableau reasoning, because the completion graphs generated by today's reasoners have a certain correspondence with models but they are not equivalent. Also, certain optimizations employed in automatic reasoners make the information found in these graphs incomplete from the point of view of model exploration. This section deals with the issues involved in using practical tableau reasoners to generate completion graphs for model exploration.

Complex Roles: the trouble with transitive roles and general role inclusion axioms (RIA) for model exploration is that tableau algorithms usually treat transitivity and RIAs rather indirectly [6–9].

Thus, completion graphs do not always contain an edge for every role relationship in the corresponding model. In order to be able to visualize models, these role relationships need to be added. This is easy for simple super-roles. We left adding roles implied by transitivity or general RIAs for future work because we wanted to test the general approach first before going into the details of how to accommodate the possibly large numbers of additional arcs in our visualization.

Blocking: tableau algorithms for DLs which lack the finite model property usually use *blocking* [10, 11] to ensure termination. A model exploration system can point out blocked nodes and blockers in the model visualization or clone blockers' successors on demand to allow infinite exploration of a model.

Caching: caching is a common optimization technique in tableau reasoners [5]; in order for it to produce unpruned completion graphs, caching needs to be disabled in a reasoner for model exploration.

Preprocessing: told cycle and synonym elimination [5] are forms of preprocessing which make a knowledge base easier to reason with. A consequence of these techniques is that, from a set of equivalent named concepts, only some may be in a completion graph node's label. The others need to be added for model exploration.

5 Related Work—Tool Support for Ontology Authors

A number of tools have been developed over the years to help users of ontologies understand them in the broader sense. This section discusses classes and examples of these tools to get a perspective of the landscape around model exploration.

We group these classes into two categories: first, and predominantly, there are *axiom inspection tools*, whose purpose it is to make the axioms of an ontology and their logical implications more accessible. Ontology editors like Swoop [12],[3]

[3] http://code.google.com/p/swoop/

Protégé 4,[4] and OBOEdit,[5] visualizations for various purposes like the ones discussed in [13], as well as justifications [14], query tools, ontology metrics analyzers etc. are examples.

Secondly, there are *model inspection tools*, which generate models for ontologies and one way or another present them to users.

An effort to generate and visualize models for OWL ontologies is described in [15], introducing the *music score notation*. This notation is in fact original, but it has the drawback of not being very easy on the screen size it uses for visualization, as the authors note.

Tweezers [16], extracts models from the Pellet DL reasoner [17] to aid a very specific way of understanding an ontology: along with its other features, model inspection is to help finding those parts of an ontology which make it costly or impossible to reason with.

Garcia et al. [18] propose a translation of ontologies from description logics into the Alloy Specification Language and to have models generated and presented by the Alloy Analyzer.

None of these model inspection tools allow users to add constraints on the models as described in Section 3 or show anything similar to the status of information we discuss in Section 3.1. The relevant papers also do not report on any empiric evidence of usefulness in general ontology engineering tasks. Only Tweezers supports infinite models by marking blocked nodes (although without pointing out the blockers).

6 Testing Usefulness

After implementing model exploration in SuperModel, we conducted a pilot user study [19] in order to get an idea of the usefulness of model exploration for the understanding of ontologies. For that study, we followed the procedure recommended by [20, Part II]. The details of our study's preparation and implementation can be found in [1].

6.1 Hypotheses

A study tests hypotheses. A user study tests hypotheses about the usability and usefulness for a particular set of applications of a piece of software. The hypotheses must be formulated such that they are testable and relevant to the software's intended use.

We first collected a number of hypotheses about what SuperModel could be useful for, and then, due to the limited time for the user study, selected the following three hypotheses:

Hyp. 1: "SuperModel can be used to understand the reason for a given entailment."

[4] http://www.co-ode.org/downloads/protege-x/
[5] http://oboedit.org/

Hyp. 2: "SuperModel can help find out, given a concept `C`, what role successors an instance of `C` must/may/may not have and which concepts they must/may/may not be instances of, and the same about their successors."

Hyp. 3: "When using SuperModel, a user can gain knowledge circumstantial to the task at hand."

The first criterion by which we selected the hypotheses to test was that the problems they proposed SuperModel facilitated solving were relevant for ontology engineering and not already sufficiently covered by already established tools. Secondly we chose hypothesis which were concrete enough and testable in an experiment. Thirdly, we selected the hypotheses for which we most expected to get positive results.

Since we also wanted to see whether model exploration was a good method to gain knowledge in these ways, and because we wanted to know, in case of negative results, whether the problem lied with model exploration or with the implementation, we tested the following hypotheses, which we hoped to refute:

Hyp. 4: "It is difficult to use model exploration to acquire information about an ontology."

Hyp. 5: "The complexity of the user interface of SuperModel is high compared to the actual tasks which it is designed to facilitate."

6.2 Participants

As participants for our study, we invited researchers in the field of ontology design and description logics. All in all, we had twelve participants, four of which rated themselves formal logic experts and experts in the use of some ontology editor, four just formal logic experts and four just ontology editor experts in the pre-test questionnaire.

It turned out that, although accidental, this composition of our population sample was fortunate as it allowed for interesting comparisions between the groups of logic experts and ontology engineers.

6.3 Experiments

We conducted our user study in two phases. The first and second phase were primarily meant to test Hyps. 2, and 1, respectively. We gathered evidence relating to Hyp. 3 and specializations of Hyps. 4 and 5 along the way. The instruments used to gather data were experimenter's observations, logging of clicks and times, and questionnaires throughout the experiments.

In the first phase, the participants were introduced to a toy ontology that was designed specifically for this user study. They were then shown how SuperModel might be used to answer a question in the style of Hyp. 2, before being asked to try and answer three similar questions about that ontology. The toy ontology and the example question were the same as the ones briefly introduced in Sec. 3.2.

In the second phase, the participants were presented justifications and asked to try to understand them. In case they gave up on understanding some justification, they were given the opportunity to try model exploration using SuperModel on the concepts occurring in the justification. The idea was to see whether a considerable amount of justifications that at first were too difficult could be understood with the help of model exploration.

In order to ensure that the problems were realistic and at the same time hard enough for participants to give up and use SuperModel sufficiently often, we used justifications that had been gathered from various published ontologies for a different user study and had there proven to be difficult to understand.

Automated click counts, experimenter's observations of frustration and misunderstandings throughout the experiment as well as answers in a post-test questionnaire were to give us information about the usability of our prototype.

6.4 Test Results

Phase 1: the first phase of the study went fairly successfully: out of the twelve participants, only two could not answer all of the questions presented to them satisfactorily. Only two reported they did not enjoy using SuperModel for the tasks presented to them in this phase of the study. All of the participants stated they could imagine asking themselves questions like the ones presented in this phase about a real-life ontology.

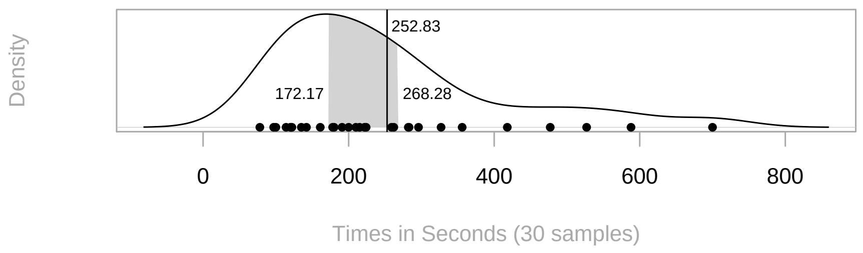

Fig. 6: Est. Density and Middle Tercile, and Average of Time needed in Phase 1

We had estimated five minutes acceptable to answer the kinds of questions we asked with an unfamiliar tool. Fig. 6 shows a distribution density function [21] estimated from our thirty samples, the gray area indicates the middle tercile of the function and the mark at 252.83s shows the average of our test results. The first two terciles are well below the mark of 300s.

Only a minority of five participants reported they would have preferred answering the questions some other way. Comparison with demographic data from the questionnaires shows that the group of those who preferred SuperModel over other methods had a higher percentage of self-reported experts in the use of ontology editors and a lower percentage of self-reported experts of formal logics. Also, the two failures to answer one of the questions asked in this phase fall into the group of logic experts who were not ontology editor experts.

Phase 2: out of 58 samples collected in Phase 2, 32 were cases in which participants initially gave up on a justification, out of which 15 were then eventually understood after activating SuperModel.

In the questionnaire following the understanding of a task after activating SuperModel, SuperModel was fully credited with giving the relevant clues only once. However, in ten cases, participants reported it gave them "some of the clues" necessary to understand the entailment.

No conclusive evidence was collected supporting Hyp. 3.

Post-Test Questionnaire: seven participants thought model exploration was definitely useful to understand entailments. Three of these were not sure. None thought either SuperModel or model exploration in general was useless and noone believed SuperModel was definitely useful but maybe model exploration was not.

Eight believed model exploration was useful for other tasks related to ontology engineering, the rest was not sure. Two participants thought SuperModel was difficult to use; three found it easy, the rest thought it normal.

6.5 Interpretation

The data for Phase 1 of the experiment suggests that the kind of question asked in it can indeed be answered in acceptable time using SuperModel. Together with the demographic data it also suggests that the benefit from SuperModel increases with familiarity with ontology editors and decreases with knowledge about formal logics.

The reason for the former could be that users of ontology editors are used to graphical environments that actively support them in the understanding of ontologies. The latter may be caused by experts of formal logics having developed their own strategies for extracting information from logical theories.

The fact that, in Phase 2, almost half of the cases where participants could not understand a justification on its own were understood with SuperModel is encouraging. However, in more than half of those cases, they spent considerably more time with SuperModel than we hoped they would, and only in two thirds of the cases they credited SuperModel with giving them at least some of the clues they needed.

We did not find any conclusive evidence supporting Hyp. 3. The reason for this may have been that the ontologies were very simple and contained little information that could have been learned in addition to what was necessary.

The measures on usability of SuperModel's UI suggest that there is room for improvement, but also that the problems are not so grave as to make it difficult to use. This is consistent with the fact that most participants thought its usability was at least 'normal'. Based on that, it seems likely that a study on an improved version of SuperModel with a more intuitive user interface would not yield drastically different results.

As the questions asked in Phase 1 were attested high relevance for everyday ontology engineering and as the justifications presented to the participants in Phase 2 were taken from various published ontologies, we hope that our results carry over to real-live activites in this area reasonably well.

7 Conclusion

In this paper, we have introduced model generation, a technique for supporting ontology engineers in their understanding of ontologies. We have explained how structures similar to models can be extracted from tableau reasoners and how the specific reasoning algorithms affect the models and additional information found in these structures. We have discussed the issues and choices involved in implementing model exploration and produced a prototype whose usefulness was assessed in a pilot user study, thus providing a first idea of the usefulness of model exploration.

The results of this study are encouraging: they suggest that model exploration can indeed help users answer certain questions about an ontology and, to a certain degree, to understand justifications.

The study has also identified starting points for future work both on our implementation and on model exploration in general.

On the Prototype: we would like to further develop SuperModel to improve usability, make it more independent of the specific reasoner used, and to enable different styles of use.

On Testing Usefulness: although it did yield valuable results, the study we report on in this paper is only a pilot study in scope. More pilot studies and experience with model exploration is needed to identify probable strengths of model exploration and ways to confirm them in an in-depth user study with more intense experiments. Apart from that, the use of mandatory information should be evaluated.

On Model Exploration: we would like to explore the possibilities and options for handling transitive roles and general role inclusion axioms. Also, it would be very interesting to investigate means to give users, for a piece of information they find while exploring a model, an explanation of why the reasoner put this information in the completion graph and, possibly, which other options there were.

References

1. Bauer, J.: Model Exploration to Support Understanding of Ontologies. Diplomarbeit, Technische Universität Dresden (2009) Advisers-Baader, F. and Lutz, C.
2. Baader, F., Nutt, W.: Basic Description Logics. [22] chapter 2
3. Horrocks, I., Hustadt, U., Sattler, U., Schmidt, R.: Computational Modal Logic. In Blackburn, P., van Benthem, J., Wolter, F., eds.: Handbook of Modal Logic. Elsevier (2006) 181–245
4. Horrocks, I.: Implementation and Optimisation Techniques. [22] chapter 9
5. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006). Volume 4130 of Lecture Notes in Artificial Intelligence., Springer (2006) 292–297
6. Horrocks, I., Sattler, U.: Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), Morgan Kaufmann, Los Altos (2003) 343–348
7. Horrocks, I., Sattler, U.: Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms. Artificial Intelligence **160**(1–2) (December 2004) 79–104
8. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), AAAI Press (2006) 57–67
9. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. J. of Automated Reasoning **39**(3) (2007) 249–276
10. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In McAllester, D., ed.: Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000). Volume 1831 of Lecture Notes in Computer Science., Springer (2000) 482–496
11. Motik, B., Shearer, R., Horrocks, I.: A Hypertableau Calculus for $\mathcal{SHIQ}$. In: Proc. of the 2007 Description Logic Workshop (DL 2007). Volume 250 of CEUR (http://ceur-ws.org/). (2007)
12. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B., Hendler, J.: Swoop: A Web Ontology Editing Browser. Journal of Web Semantics **4** (2005) 2005
13. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods—a survey. ACM Comput. Surv. **39**(4) (2007) 10
14. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. PhD thesis, University of Maryland at College Park, College Park, MD, USA (2006) Adviser-James Hendler.
15. Barinskis, M., Barzdins, G.: Satisfiability Model Visualization Plugin for Deep Consistency Checking of OWL Ontologies. In Golbreich, C., Kalyanpur, A., Parsia, B., eds.: OWLED. Volume 258 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
16. Wang, T., Parsia, B.: Ontology Performance Profiling and Model Examination: First Steps. In Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Schreiber, G., Cudré-Mauroux, P., eds.: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea. Volume 4825 of LNCS., Berlin, Heidelberg, Springer Verlag (November 2007) 589–602

17. Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics **5**(2) (2007) 51–53
18. Garcia, M., Kaplunova, A., Möller, R.: Model Generation in Description Logics: What Can We Learn From Software Engineering? Technical report, Institute for Software Systems (STS), Hamburg University of Technology, Germany (2007) See http://www.sts.tu-harburg.de/tech-reports/papers.html.
19. Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1997)
20. Dumas, J.S., Redish, J.C.: A Practical Guide to Usability Testing. Intellect Books, Exeter, UK (1999)
21. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2008) ISBN 3-900051-07-0.
22. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)