Towards Reasoning Pragmatics

Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton, Ohio
http://www.pascal-hitzler.de/

Abstract. The realization of Semantic Web reasoning is central to substantiating the Semantic Web vision. However, current mainstream research on this topic faces serious challenges, which force us to question established lines of research and to rethink the underlying approaches.

1 What is Semantic Web Reasoning?

The ability to combine data, mediated by metadata, in order to derive knowledge which is only implicitly present, is central to the Semantic Web idea. This process of accessing implicit knowledge is commonly called *reasoning*, and formal model-theoretic semantics tells us exactly what knowledge is implicit in the data.[1]

Let us attempt to define reasoning in rather general terms: *Reasoning is about arriving at the exact answer(s) to a given query.* Formulated in this generality, this encompasses many situations which would classically not be considered reasoning – but it will suffice for our purposes. Note that the definition implicitly assumes that there *is* an exact answer. In a reasoning context, such an exact answer would normally be defined by a model-theoretic semantics.[2]

Current approaches to Semantic Web reasoning, however, which are mainly based on calculi drawn from predicate logic proof theory, face several serious obstacles.

- Scalability of algorithms and systems has been improving drastically, but systems are still incapable of dealing with amounts of data on the order of magnitude as can be expected on the World Wide Web. This is aggrevated by the fact that classical proof theory does not readily allow for parallelization, and that the amount of data present on the web increases with a similar growth rate as the efficiency of hardware.
- Realistic data, in particular on the web, is generally noisy. Established proof-theoretic approaches (even those including uncertainty or probabilistic methods) are unable to cope with this kind of data in a manner which is ready for large-scale applications.

[1] It is rather peculiar that a considerable proportion of so-called Semantic Web research and publications ignores formal semantics. Even most textbooks fail to explain it properly. An exception is [7].

[2] Simply referring to a *formal* semantics is too vague, since this would also include procedural semantics, i.e. non-declarative approaches, and thus would include most mainstream programming languages.

– It is a huge engineering effort to create web data and ontologies which are of sufficiently high quality for current reasoning approaches, and usually beyond the abilities of application developers. The resulting knowledge bases are furthermore severely limited in terms of reusability for other application contexts.

The state of the art shows no indications that approaches based on logical proof theory would overcome these obstacles anytime soon in such a way that large-scale applications on the web can be realized. Since reasoning is central to the Semantic Web vision, we are forced to rethink our traditional methods, and should be prepared to tread new paths.

A key idea to this effect, voiced by several researchers (see e.g. [3, 23]) is to explore alternative methods for reasoning. These may still be based more or less closely on proof-theoretic considerations, or they may not. They could, e.g., utilize methods from statistical machine learning or from nature-inspired computing.

Researchers who are used to thinking in classical proof-theoretic terms are likely to object to this thought, arguing that a relaxation of strict proof-theoretic requirements on algorithms, such as soundness and completeness, would pave the way for arbitrary algorithms which do not perform logical reasoning at all, and thus would fail to adhere to the specification provided by the formal semantics underlying the data – and thus jeopardize the Semantic Web vision. While such arguments have some virtue, it needs to be stressed that the nature of the underlying algorithm is, effectively, unimportant, as long as the system adheres to the specification, i.e. to the formal semantics.

Imagine, as a thought experiment, a black box system which performs sound and complete reasoning in all application settings it is made for – or at least up to the extent to which standard reasoning systems are sound and complete.[3] Does it matter then whether the underlying algorithm is *provably* sound and complete? I guess not. The only important thing is that its *performance* is sound and complete. If the black box were orders of magnitude faster then conventional reasoners, but somebody would tell you that it is based on statistical methods, which one would you choose to work with? Obviously, the answer depends on the application scenario – if you'd like to manage a bank account, you may want to stick with the proof-theoretic approach since you can prove formally that the algorithm does what it should; but if you use the algorithm for web search, the quicker algorithm might be the better choice. Also, your choice will likely depend on the evidence given as to the correctness of the black box algorithm in application settings.

This last thought is important: If a reasoning system is not based on proof theory, then there must be a quality measure for the system, i.e., the system must

[3] Usually, they are not sound and complete, although they are based on underlying algorithms which are, theoretically, sound and complete. Incompleteness comes from the fact that resources, including time, are limited. Unsoundness comes from bugs in the system.

be evaluated against the gold standard, which is given by the formal semantics, or equivalently by the provably sound and complete implementations [23].

If we bring noisy data, as on the web, into the picture, it becomes even clearer why a fixation on soundness and completeness of reasoning systems is counterproductive for the Semantic Web: In the presence of such data, even the formal model-theoretic semantics breaks down, and it is quite unclear how to develop algorithms based on proof theory for such data. The notions of soundness and completeness of reasoning in the classical sense appear to be almost meaningless. But only almost, since alternative reasoning systems which are able to cope with noisy data can still be evaluated against the gold standard on non-noisy data, for quality assurance.

In the following, we revisit the role of soundness and completeness for reasoning, and argue further for an alternative perspective on these issues (Section 2). We also discuss key challenges which need to be addressed in order to realise reasoning on and for the Semantic Web, in particular the questions of expressivity of ontology languages (Section 3), roads to bootstrapping (Section 4), knowledge acquisition (Section 5), and user interfacing (Section6). We conclude in Section 7.

2 The Role of Soundness, Completeness, and Computational Complexity

Computational complexity has classically been a consideration for the development of description logics, which underlie the Web Ontology Language OWL – which is currently the most prominent ontology language for Semantic Web reasoning. In particular, OWL is a decidable logic. The currently ongoing revision OWL 2 [6] furthermore explicitly defines fragments, called profiles, with lower (in fact, polynomial) computational complexity.

Soundness and completeness are central properties of classical reasoning algorithms for logic-based knowledge representation languages, and are thus central notions in the development of Semantic Web reasoning around OWL. However performance issues have prompted researchers to advocate *approximate reasoning* for the Semantic Web (see e.g. [3, 23]). Arguing for this approach provokes radically different kinds of reactions: some logicians appear to be abhorred by the mere thought, while many application developers find it the most natural thing to do. Often it turns out that the opposing factions misunderstand the arguments: counterarguments usually state that leaving the model-theoretic semantics behind would lead to arbitrariness and thus loss of quality. So let it be stated again explicitly: approximate reasoning shall not replace sound and complete reasoning in the sense that the latter would no longer be needed. Quite in contrast, approximate reasoning in fact needs the sound and complete approaches as a gold standard for evaluation and quality assurance. The following shall help to make this relationship clear.

2.1 Sound but Incomplete Reasoning

There appears to be not much argument against this in the Semantic Web commmunity, even from logicians: they are used to this, since some KR languages, including first-order predicate logic, are only semi-decidable,[4] i.e. completeness can only be achieved with unlimited time resources anyway. For decidable languages, however, a sound but incomplete reasoner should always be evaluated against the gold standard, i.e., against a sound and complete reasoner, in order to show the amount of incompleteness incurred versus the gain in efficiency. Interestingly, this is rarely done in a structured way, which is, in my opinion, a serious neglect. A statistical framework for evaluation against the gold standard is presented in [23].

2.2 Unsound but Complete Reasoning

Allowing for reasoning algorithms to be unsound appears to be much more controversial, and the usefulness of this concept seems to be harder to grasp. However, there are obvious examples. Consider, e.g., fault-detection in a power plant in case of an emergency: The system shall determine (quickly!) which parts of the factory need to be shut down. Obviously, it is of highest importantance that the critical part is contained in the shutdown, while it is less of a problem if too many other parts are shut down, too.[5] Another obvious example is semantic search: In most cases, users would prefer to get a quick set of replies, among which the correct one can be found, rather than wait longer for one exact answer.

Furthermore, sound-incomplete and unsound-complete systems can sometimes be teamed up and work in parallel to provide better overall performance (see e.g. [24]).

2.3 Unsound and Incomplete Reasoning

Following the above arguments to their logical conclusion, it should become clear why unsound and incomplete reasoning has its place among applications. Remember that there is the gold standard against which such systems should be evaluated. And obviously there is no reason to stray from a sound and complete approach if the knowledge base is small enough to allow for it.

The most prominent historic example for an unsound and incomplete yet very successful reasoning system is Prolog. Traditionally, the unification algorithm, which is part of the SLD-resolution proof procedure used in Prolog [13], is used without the so-called *occurs check*, which, depending on the exact implementation, can cause unsoundness [16].[6] This omission was made due to reasons

[4] Some non-montonic logics are not even semi-decidable.

[5] This example is due to Frank van Harmelen, personal communication.

[6] To obtain a wrong answer, execute the query `?-p(a).` on the logic program consisting of the two clauses `p(a) :- q(X,X).` and `q(X,f(X)).`, e.g. under SWI-Prolog. – The example is due to Markus Krötzsch.

of efficiency, and turned out to be feasible since it rarely causes a problem for Prolog programmers.

Likewise, it is not unreasonable to expect that carefully engineered unsound and incomplete reasoning approaches can be useful on the Semantic Web, in particular when sound and complete systems fail to provide results within a reasonable time span. Furthermore, there is nothing wrong with using entirely alternative approaches to this kind of reasoning, e.g., approaches which are not based on proof theory.

To give an example of the latter, we refer to [2], where the authors use a statistical learning approach using support vector machines. They train their machine to infer class membership in $\mathcal{ALC}$, which is a description logic related to OWL, and achieve a 90% coverage. Note that this is done without any proof theory, other than to obtain the training examples. In effect, their system learns to reason with high coverage without performing logical deduction in the proof-theoretic sense.

For a statistical framework for evaluation against the gold standard we refer again to [23].

2.4 Computational Complexity and Decidability

Considerations on computational complexity and decidability have been driving research around description logics, which underlie OWL, from the beginning. At the same time, there are more and more critical voices concerning the fixation of that research on these issues, since it is not quite clear how practical systems would benefit from these. Indeed, theoretical (worst-case) computational complexity is hardly a relevant measure for the performance of real systems. At the same time, decidability is only guaranteed assuming bug-free implementations– which is an unrealistic assumption –, and given enough resources – which is also unrealistic since the underlying algorithms often require exponential time in the worst case.

The misconception underlying these objections is that computational complexity and decidability are not practial measures which have a direct meaning in application contexts. They are rather a priori measures for language and algorithm development, and the recent history of OWL language development indicates that these a priori measures have indeed done a good job. It is obviously better to have such theoretical means for the conceptual work in creating language features, than to have no measures at all. And indeed this has worked out well, since e.g. reasoning systems based on realistic OWL knowledge bases currently seem to behave rather well despite the high worst-case computational complexity.

Taking the perspective of approximate reasoning algorithms as laid out earlier, it is actually a decisively positive feature of Semantic Web knowledge representation languages that systems exist which can serve as a gold standard reference. Considering the difficulties in other disciplines (like Information Retrieval) in creating gold standards, we indeed are delivered the gold standard on a silver plate. We can use this to an advantage.

3 Diverse Knowledge Representation Issues

Within 50 years of KR research, many issues related to the representation of non-classical knowledge have been investigated. Many of the research results obtained in this realm are currently carried over to ontology languages around OWL, including abductive reasoning, uncertainty handling, inconsistency handling and paraconsistent reasoning, closed world reasoning and non-monotonicity, belief revision, etc.

However all these approaches face the same problems that OWL reasoning faces, foremost scalability and the dealing with realistic noisy data.[7] Indeed under most of these approaches, runtime performance becomes worse, since the reasoning problems generally become harder in terms of computational complexity.

Nevertheless, research on logical foundations of these knowledge representation issues, as currently being carried out, is needed to establish the gold standard. At this time there is a certain neglect, however, in combining several paradigms, e.g. it is quite unclear how to marry paraconsistent reasoning with uncertainty handling.

Research into enhancing expressivity of ontology languages can roughly be divided into the following.

- Classical logic features: This line of research follows the well-trodden path of extending e.g. OWL with further expressive features, while attempting to retain decidability and in some cases low computational complexity. Some concrete suggestions for next steps in this direction are given in the appendix.
- Extralogical features: These include datatypes and additional datastructures, like e.g. Description Graphs [19].
- Supraclassical logic: Logical features related to commonsense reasoning like abduction and explanations (e.g., [8]), paraconsistency (e.g., [15]), belief revision, closed-world (e.g., [4]), uncertainty handling (e.g., [10, 14]), etc. There is hardly any work investigating approximate reasoning solutions for supraclassical logics.

Investigations into these issues should first establish the gold standard following sound logical principles including computational complexity issues. Only then should extensions towards approximate reasoning be done.

4 Bootstrapping Reasoning

How to get from A (today) to B (reasoning that works on the Semantic Web)? I believe that a promising approach lies in bootstrapping existing applications which use little or no reasoning, based e.g. on RDF. The idea is to enhance

[7] I'm personally critical about fuzzy logic and probabilistic logic approaches in practice for Semantic Web issues. Dealing with noisy data on the web does not seem to easily fall in the fuzzy or probabilistic category. So probably new ideas are needed for these.

these applications very carefully with a bit more reasoning, in order to clearly understand the added value and the difficulties one is facing when doing this. A (very) generic workflow for the bootstrapping may be as follows.

1. Identify an (RDF) application where enhanced reasoning would bring added value.
2. Identify ontology language constructs which would be needed for expressing the knowledge needed for the added value.
3. Identify an ontology language (an OWL profile or an OWL+Rules hybrid) which covers these additional language constructs.
4. Find a suitable reasoner for the enhanced language.
5. Enhance the knowledge base and plug the software components together.

The point of these exercises is not only to show that more reasoning capabilities bring added value, but also to identify obstacles in the bootstrapping process.

5 Overcoming The Ontology Acquisition Bottleneck

The ontology acquisition bottleneck for logically expressive ontologies comes partly from the fact that sound and complete reasoning algorithms work only on carefully devised ontologies, and in many cases it needs an ontology expert to develop them. Creating such high-quality ontologies is very costly.

A partial solution to this problem is related to (1) noise handling and (2) the bootstrapping idea. With the current fixation on sound and complete reasoning it cannot be expected that usable ontologies (in the sound and complete sense) will appear in large quantities e.g. on the web. However, it is conceivable that e.g. Linked Open Data[8] (LoD) could be augmented with more expressive schema data to allow e.g. for reasoning-based semantic search. The resulting extended LoD cloud would still be noisy and not readily usable with sound and complete approaches. So reasoning approaches which can handle noise are needed. This is also in line with the bootstrapping idea: We already have a lot of metadata available, and in order to proceed we need to make efforts to enhance this data, and to find robust reasoning techniques which can deal with this real-world noisy data.

6 Human Interfacing

Classical ontology engineering often has the appearance of expert sysem creation, if used off the web. On the web, it often lacks the reasoning component.

As argued in this paper, in order to advance the Semantic Web vision – on and off the web – we need to find ways to reason with noisy and incomplete data in a realistic and pragmatic manner. This necessity is not reflected by current ontology engineering research.

[8] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/

In order to advance towards ontology reasoning applications, we need ontology engineering systems which

- support reasoning bootstrapping,
- include multiple reasoning support (i.e. multiple reasoning algorithms, classical and non-classical), and
- are made to cope with noisy and uncertain data.

We need to get away from thinking about ontology creation as coding in the programming sense. This can only be achieved by relaxing the reasoning algorithm requirements, i.e. by realising reasoning systems which can cope with noisy and uncertain data.

7 Putting It All Together

In this paper I argue for roads to realising reasoning on and for the Semantic Web. Efforts on several fronts are put forth:

- The excellent research results and ongoing efforts in establishing sound and complete proof-theory-based reasoning systems need to be complemented by investigations into alternative reasoning approaches, which are not necessarily based on proof theory, and can handle noisy and uncertain data.
- Reasoning bootstrapping should be investigated seriously and on a broad front, in order to clearly show added value, and in order to identify challenges in adopting ontology reasoning in applications.
- Ontology engineering environments should systematically accommodate reasoning bootstrapping and the support of multiple reasoning paradigms, classical and non-classical.

Let us recall a main point of this paper: reasoning algorithms do not have to be based on proof theory. But they have to perform well if compared with the gold standard.

In a sense, the laid out lines of research lead us a bit further away from knowledge representation (KR), and at the same time they do a small step towards non-KR-based intelligent systems: Not all the intelligence must be in the knowledge base (with corresponding sound and complete reasoning algorithms). We must facilitate intelligent solutions, including machine learning and data mining, and statistical inductive methods, to achieve the Semantic Web vision.

Acknowledgement

Many thanks to Cory Henson, Prateek Jain, and Valentin Zacharias for feedback and discussions. The appendix is taken from [5].

References

1. H. Boley and M. Kifer, editors. *RIF Framework for Logic Dialects*. W3C Working Draft, 30 July 2008. Available at http://www.w3.org/TR/rif-fld/.
2. N. Fanizzi, C. d'Amato, and F. Esposito. Statistical learning for inductive query answering on owl ontologies. In A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. W. Finin, and K. Thirunarayan, editors, *The Semantic Web – ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*, pages 195–212. Springer, 2008.
3. D. Fensel and F. van Harmelen. Unifying reasoning and search to web scale. *IEEE Internet Computing*, 11(2):96, 94–95, 2007.
4. S. Grimm and P. Hitzler. A preferential tableaux calculus for circumscriptive ALCO. In *Proceedings of the Third International Conference on Web Reasoning and Rule Systems, Washington D.C., USA, October 2009*, Lecture Notes in Computer Science. Springer, 2009. To appear.
5. P. Hitzler. Suggestions for OWL 3. In *Proceedings of OWL – Experiences and Directions, Sixth International Workshop, Washington D.C., USA, October 2009*, 2009. To appear.
6. P. Hitzler, M. Krötzsch, B. Parsia, P.F. Patel-Schneider, and S. Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Proposed Recommendation, 22 September 2009. Available at http://www.w3.org/TR/2009/PR-owl2-primer-20090922/.
7. P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.
8. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in owl. In A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. W. Finin, and K. Thirunarayan, editors, *The Semantic Web – ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*, pages 323–338. Springer, 2008.
9. I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *SWRL: A Semantic Web Rule Language*. W3C Member Submission, 21 May 2004. Available at http://www.w3.org/Submission/SWRL/.
10. P. Klinov and B. Parsia. Optimization and evaluation of reasoning in probabilistic description logic: Towards a systematic approach. In A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. W. Finin, and K. Thirunarayan, editors, *The Semantic Web – ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*, pages 213–228, 2008.
11. M. Krötzsch, S. Rudolph, and P. Hitzler. Description logic rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008*, pages 80–84. IOS Press, 2008.
12. M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayan, editors, *Proceedings of the 7th International Semantic Web Conference (ISWC-08)*, volume 5318 of *Lecture Notes in Computer Science*, pages 649–664. Springer, 2008.
13. John W. Lloyd. *Foundations of Logic Programming*. Springer, 1987.

14. T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal on Web Semantics*, 6(4):291–308, 2008.
15. Y. Ma and P. Hitzler. Paraconsistent reasoning for OWL 2. In *Proceedings of the Third International Conference on Web Reasoning and Rule Systems, Washington D.C., USA, October 2009*, Lecture Notes in Computer Science. Springer, 2009. To appear.
16. K. Marriott and H. Sondergaard. On Prolog and the occur check problem. *SIGPLAN Not.*, 24(5):76–82, 1989.
17. D.L. McGuinness and F. van Harmelen, editors. *OWL Web Ontology Language Overview*. W3C Recommendation, 10 February 2004. Available at http://www.w3.org/TR/owl-features/.
18. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, editors. *OWL 2 Web Ontology Language: Profiles*. W3C Proposed Recommendation, 22 September 2009. Available at http://www.w3.org/TR/2009/PR-owl2-profiles-20090922/.
19. B. Motik, B. Cuenca Grau, I. Horrocks, and U. Sattler. Representing Ontologies Using Description Logics, Description Graphs, and Rules. *Artificial Intelligence*, 173(14):1275–1309, 2009.
20. B. Motik, P.F. Patel-Schneider, and B. Parsia, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. W3C Candidate Recommendation, 22 September 2009. Available at http://www.w3.org/TR/2009/PR-owl2-syntax-20090922/.
21. B. Motik, U. Sattler, and R. Studer. Query-answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
22. S. Rudolph, M. Krötzsch, and P. Hitzler. Cheap Boolean role constructors for description logics. In S. Hölldoble, C. Lutz, and H. Wansing, editors, *Proceedings of 11th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 5293 of *Lecture Notes in Artificial Intelligence*, pages 362–374. Springer, 2008.
23. S. Rudolph, T. Tserendorj, and P. Hitzler. What is approximate reasoning? In D. Calvanese and G. Lausen, editors, *Web Reasoning and Rule Systems, Second International Conference, RR 2008, Karlsruhe, Germany, October 31-November 1, 2008. Proceedings*, volume 5341 of *Lecture Notes in Computer Science*, pages 150–164, 2008.
24. T. Tserendorj, S. Rudolph, M. Krötzsch, and P. Hitzler. Approximate owl-reasoning with screech. In D. Calvanese and G. Lausen, editors, *Web Reasoning and Rule Systems, Second International Conference, RR 2008, Karlsruhe, Germany, October 31-November 1, 2008. Proceedings*, volume 5341 of *Lecture Notes in Computer Science*, pages 165–180, 2008.
25. W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Working Draft, 22 September 2009. Available at http://www.w3.org/TR/2009/PR-owl2-overview-20090922/.

A Appendix: Suggestions for OWL 3

Abstract. With OWL 2 about to be completed, it is the right time to start discussions on possible future modifications of OWL. We present here a number of suggestions in order to discuss them with the OWL user community. They encompass expressive extensions on polynomial OWL 2 profiles, a suggestion for an OWL Rules language, and expressive extensions for OWL DL.

A.1 Introduction

The OWL community has grown with breathtaking speed in the last couple of years. The improvements coming from the transition from OWL 1 [17] to OWL 2 [25] are an important contribution to keeping the language alive and in synch with the users. While the standardization process for OWL 2 is currently coming to a successful conclusion, it is important that the development process does not stop, and that discussions on how to improve the language continue.

In this appendix, we present a number of suggestions for improvements to OWL DL,[9] which are based on some recent work. We consider it important that such further development is done in alignment with the design principles of OWL, and in particular with the description logic perspective which has governed its creation. Indeed, this heritage has been respected in the development of OWL 2, and is bringing it to a fruitful conclusion. There is no apparent reason for straying from this path.

In particular, the following general rationales should be adhered to, as has happened for OWL 1 and OWL 2.

- Decidability of OWL DL should be retained.
- OWL DL semantics should be based on a first-order predicate logic semantics (and as such should, in particular, be essentially open-world and monotonic).
- Analysis of computational complexities shall govern the selection of language features in OWL DL.

Obviously, there are other important issues, like basic compatibility with RDF, having an XML-based syntax, backward-compatibility, etc., but we take these for granted and do not consider them to be major obstacles as long as future extensions of OWL are developed along the inherited lines of thinking.

The suggestions which we present below indeed adhere to the design rationales just laid out. They concern different aspects of the language, and are basically independent of each other, i.e. they can be discussed separately. At the same time, however, they are also closely related and compatible, so that it is reasonable to discuss them together.

[9] OWL DL has always played a special role in defining OWL – it is the basis from which OWL Full and other variants, like OWL Lite or the OWL 2 profiles, are developed. So in this appendix we focus on OWL DL.

In Section A.2, we suggest a rule-based syntax for OWL. The syntax is actually of a hybrid nature, and allows e.g. class descriptions inside the rules. Nevertheless, it captures OWL with a syntax which is essentially a rule-syntax.

In Section A.3, we suggest the addition of Boolean role expressions to the arsenal of language constructs available in OWL. We also explain which cautionary measures need to be taken when this is done, in order to not lose decidability and complexity properties.

In Section A.4, we suggest considerably extending OWL by including the DL-safe variable fragment of SWRL [9] together with the DL-safe fragment [21] of SWRL.

In Section A.5, we propose a tractable profile, called ELP, which encompasses OWL 2 EL, OWL 2 RL, most of OWL 2 QL, and some expressive means which are not contained in OWL 2. It is currently the most expressive polynomial language which extends OWL 2 EL and OWL 2 RL, and is still relatively easy to implement.

In Section A.6, we conclude.

Obviously, we do not have the space to define all these extensions in detail, or to discuss all aspects of them exhaustively. We thus strive to convey the main ideas and intuitions, and refer to the indicated literature for details. In the definitions and discussions, we will sometimes drop details, or remain a bit vague (and thus compromise completeness of our exhibition), in order to be better able to focus on the main arguments. We believe that this serves the discussion better than being entirely rigorous on the formal aspects.

A.2 An OWL Rules Language

The alignment of rule languages with OWL (and vice versa) has been a much (and sometimes heatedly) discussed topic. The OWL paradigm is quite different in underlying intuition, modelling style, and expressivity than standard rule language paradigms. Recent efforts involving OWL and rules attempt to merge the paradigms in order to get the best of both worlds.

The advance from OWL 1 to OWL 2 indeed brings the two paradigms closer together. In particular, a considerable variety of rules, understood as Datalog rules with unary and binary predicates under a first-order predicate logic semantics, can be translated with some effort directly into OWL 2 DL. This observation paves the way for a rule-based syntax for OWL, which we will briefly present below. The suggestions in this section are based on [11].

Given any description logic D, a D-rule is a rule of the form

$$A_1 \wedge \cdots \wedge A_n \rightarrow A,$$

where A and A_i are expressions of the form $C(x)$ or $R(x, y)$, where C are (possibly non-atomic) concept expressions over D, R are role names (or role expressions if allowed in D), and x, y are either variables or individual names (y may also be a datatype value if this is allowed in D), and the following conditions are satisfied.

$$\begin{aligned}
\mathrm{Man}(x) \wedge \mathrm{hasBrother}(x, y) \wedge \mathrm{hasChild}(y, z) &\rightarrow \mathrm{Uncle}(x) \\
\mathrm{ThaiCurry}(x) &\rightarrow \exists \mathrm{contains.FishProduct}(x) \\
\mathrm{kills}(x, x) &\rightarrow \mathrm{PersonCommittingSuicide}(x) \\
\mathrm{PersonCommittingSuicide}(x) &\rightarrow \mathrm{kills}(x, x) \\
\mathrm{NutAllergic}(x) \wedge \mathrm{NutProduct}(y) &\rightarrow \mathrm{dislikes}(x, y) \\
\mathrm{dislikes}(x, z) \wedge \mathrm{Dish}(y) \wedge \mathrm{contains}^{-}(z, y) &\rightarrow \mathrm{dislikes}(x, y) \\
\mathrm{worksAt}(x, y) \wedge \mathrm{University}(y) \wedge & \\
\wedge \mathrm{supervises}(x, z) \wedge \mathrm{PhDStudent}(z) &\rightarrow \mathrm{professorOf}(x, z) \\
\mathrm{Mouse}(x) \wedge \exists \mathrm{hasNose.TrunkLike}(y) &\rightarrow \mathrm{smallerThan}(x, y)
\end{aligned}$$

Fig. 1. A $\mathcal{SROIQ}$-Rules knowledge base.

- The pattern of variables in the rule body forms a tree. This is to be understood in the sense that whenever there is an expression $R(x, y)$ with a role R and two variables x, y in the rule body, then there is a directed edge from x to y – hence each body gives rise to a directed graph, and the condition states that this graph must be a tree. Note that individuals are not taken into account when constructing the graph.[10] Note also that the rule body must form a single tree.
- The first argument of A is the root of the just mentioned tree.

Semantically, $\mathcal{SROIQ}$-rules come with the straightforward meaning under a first-order predicate logic reading, i.e., the implication arrow is read as first-order implication, and the free variables are considered to be universally quantified.

A D-Rules knowledge base consists of a (finite) set of D-rules,[11] which satisfies additional constraints, which depend on D. These constraints guarantee that certain properties of D, e.g., decidability, are preserved.

For OWL 2 DL, these additional constraints specify regularity conditions and restrictions on the use of non-simple roles, similarly to $\mathcal{SROIQ}(D)$ – we omit the details. Examples for $\mathcal{SROIQ}$-rules are given in Figure 1.

The beauty of $\mathcal{SROIQ}$-rules lies in the fact that any $\mathcal{SROIQ}$-Rules knowledge base can be transformed into a $\mathcal{SROIQ}$ knowlege base – and that the transformation algorithm is polynomial. This means that $\mathcal{SROIQ}$-rules are nothing more or less than a sophisticated kind of syntactic sugar for $\mathcal{SROIQ}$. It is easy to see that, in fact, any $\mathcal{SROIQ}$-axiom can also be written as a $\mathcal{SROIQ}$-rule, so that modelling in $\mathcal{SROIQ}$ can be done entirely within the $\mathcal{SROIQ}$-Rules paradigm.

In order to be a useful language, it is certainly important to develop convenient web-enabled syntaxes. Such a syntax could be based on the Rule Interchange Format (RIF) [1], for example, which is currently in the final stages

[10] The exact definition is a bit more complicated; see [11].
[11] Notice the difference in spelling: uppercase vs. lowercase.

$$\exists(\text{testifiesAgainst} \sqcap \text{relativeOf}).\top \sqsubseteq \neg\text{UnderOath}$$
$$\text{hasParent} \sqsubseteq \text{hasFather} \sqcup \text{hasMother}$$
$$\text{hasDaughter} \sqsubseteq \text{hasChild} \sqcap \neg\text{hasSon}$$

Fig. 2. Examples for Boolean role constructors

of becoming a W3C Recommendation. A $\mathcal{SROIQ}$-Rules syntax could also be defined as a straightforward extension of the OWL 2 Functional Style Syntax [20].

Proposal: OWL 3 should have a rule-based syntax based on Description Logic Rules.

A.3 Boolean Role Constructors

Boolean role constructors, i.e., conjunction, disjunction, and negation for roles, can be added to description logics around OWL under certain restrictions, without compromising language complexity. Since they provide additional modelling features which are clearly useful in the right circumstances (see Figure 2), there is no strong reason why they shouldn't be added to OWL. The following summarizes results from [22].

All Boolean role constructors can be added to $\mathcal{SROIQ}$ without compromising its computational complexity, as long as the constructors involve only simple roles – the resulting description logic is denoted by $\mathcal{SROIQB}_{\text{S}}$.

Likewise, OWL 2 EL can be extended with role conjunction without losing polynomial complexity of the language. Regularity requirements coming from $\mathcal{SROIQ}$ can be dropped (they are also not needed for polynomiality of the description logic $\mathcal{EL}^{++}$, which is well-known). Likewise, the extension of OWL 2 RL with role conjunctions is still polynomial.

While the complexity results just given are favorable, it has to be noted that suitable algorithms for reasoning with $\mathcal{SROIQB}_{\text{S}}$ still need to be developed. Algorithms for the respective extensions of OWL 2 EL and OWL 2 RL, however, can easily be obtained by adjusting known algorithms for these languages – see also Section A.5.

Proposal: OWL 3 should allow the use of Boolean role constructors wherever appropriate.

A.4 DL-safe Variable SWRL

SWRL [9] is a very natural extension for description logics with first-order predicate logic rules. Despite being a W3C Member Submission rather than a W3C Recommendation, it has achieved an extremely high visibility. However, in its

original form, SWRL is undecidable, i.e., it does not closely follow the design guidelines we have listed in the introduction.

A remedy for the decidability issue is the restriction of SWRL rules to so-called *DL-safe rules* [21]. Syntactically, DL-safe rules are rules of the form

$$A_1 \wedge \cdots \wedge A_n \rightarrow A$$

as in Section A.2, but without the requirements on tree-shapedness. Semantically, however, they are read as first-order predicate logic rules, but with the restriction that variables in the rules may bind only to individuals which are present in the knowledge base.[12] In essence, this limits the usability of DL-safe SWRL to applications which do not involve TBox reasoning.

It is now possible to generalize DL-safe SWRL without compromising decidability. The underlying idea has been spelled out in a more limited setting in [12] (see also Section A.5), but it obviously carries over to $\mathcal{SROIQ}$.

In order to understand the generalization, we need to return to $\mathcal{SROIQ}$-rules as defined in Section A.2. Recall that the tree-shapedness of the rule bodies is essential, but that role expressions involving individuals are ignored when checking for the tree structure.

The idea behind DL-safe variable SWRL is now to identify those variables in rule bodies which violate the required tree structure, and to define the semantics of the rules such that these variables may only bind to individuals present in the knowledge base – these variables are called *DL-safe variables*. The other variables are interpreted as usual under the first-order predicate logic semantics.

An alternative way to describe the same thing is to say that a rule qualifies as DL-safe variable SWRL if replacing all DL-safe variables in the rule by individuals results in an allowed $\mathcal{SROIQ}$-rule.

As an example, consider the rule

$$C(x) \wedge R(x, w) \wedge S(x, y) \wedge D(y) \wedge T(y, w) \rightarrow V(x, y),$$

which violates the requirement of tree-shapedness because there are two different paths from x to w. Now, if we replace w by an individual, say o, then the resulting rule

$$C(x) \wedge R(x, o) \wedge S(x, y) \wedge D(y) \wedge T(y, o) \rightarrow V(x, y)$$

[12] The original definition is different, but equivalent. It required that each variable occurred in an atom in the rule body, which is not an atom of the underlying description logic knowledge base. The usual way to achieve this is by introducing an auxiliary class O which contains all known individuals, and adding $O(x)$ to each rule body, for each variable in the rule. Our definition instead employs a redefinition of the semantics, which appears to be more natural in this case. Essentially, the two formulations are equivalent.

is a $\mathcal{SROIQ}$-rule.[13] Hence, the rule

$$C(x) \wedge R(x, w_s) \wedge S(x, y) \wedge D(y) \wedge T(y, w_s) \rightarrow V(x, y),$$

where w_s is a DL-safe variable, is a DL-safe variabe SWRL rule. Note that the other variables can still bind to elements whose existence is guaranteed by the knowledge base but which are not present in the knowledge base as individuals, which would not be possible if the rule were interpreted as DL-safe.

In principle, naive implementations of this language could work with multiple instantiations of rules containing DL-safe variables, but no implementations yet exist. In principle, they should not be much more difficult to deal with than DL-safe SWRL rules.

Proposal: OWL 3 DL should incorporate DL-safe SWRL and DL-safe variable SWRL.

A.5 Pushing The Tractable Profiles

The OWL 2 Profiles document [18] describes three designated profiles of OWL 2, known as OWL 2 EL, OWL 2 RL, and OWL 2 QL. These three languages have been designed with different design principles in mind. They correspond to different description logics, have different expressive features, and can be implemented using different methods.

The three profiles have in common that they are all of polynomial complexity, i.e., they are rather inexpressive languages, despite the fact that they have already found applications. While having three polynomial profiles is fine due to their fundamental differences, the question about maximal expressivity while staying in polynomial time naturally comes into view.

The ELP language [12] is a language with polynomial complexity which properly contains both OWL EL and OWL RL. It also contains most of OWL QL.[14] Furthermore, it still features rather simple algorithms for reasoning implementations.

More precisely, ELP has the following language features.

- It contains OWL 2 EL Rules, i.e. $\mathcal{EL}^{++}$-rules as defined in Section A.2.[15] Note that $\mathcal{EL}^{++}$-rules cannot be converted to $\mathcal{EL}^{++}$ (i.e. OWL 2 EL) using the algorithm which converts $\mathcal{SROIQ}$-rules to $\mathcal{SROIQ}$.

[13] This rule can be expressed in $\mathcal{SROIQ}$ by the knowledge base consisting of the three statements

$$\begin{array}{ll} C \sqcap \exists R.\{o\} \sqsubseteq \exists R_1.\text{Self} & \\ D \sqcap \exists T.\{o\} \sqsubseteq \exists R_2.\text{Self} & \text{and} \\ R_1 \circ S \circ R_2 \sqsubseteq V. & \end{array}$$

See [11].

[14] Role inverses cannot be expressed in ELP.

[15] $\mathcal{EL}^{++}$-rules are D-rules with $D = \mathcal{EL}^{++}$.

$$
\begin{aligned}
\text{NutAllergic}(x) \wedge \text{NutProduct}(y) &\rightarrow \text{dislikes}(x, y) \\
\text{Vegetarian}(x) \wedge \text{FishProduct}(y) &\rightarrow \text{dislikes}(x, y) \\
\text{orderedDish}(x, y) \wedge \text{dislikes}(x, y) &\rightarrow \text{Unhappy}(x) \\
\text{dislikes}(x, v_s) \wedge \text{Dish}(y) \wedge \text{contains}(y, v_s) &\rightarrow \text{dislikes}(x, y) \\
\text{orderedDish}(x, y) &\rightarrow \text{Dish}(y) \\
\text{ThaiCurry}(x) &\rightarrow \text{contains}(x, \text{peanutOil}) \\
\text{ThaiCurry}(x) &\rightarrow \exists \text{contains.FishProduct}(x) \\
&\rightarrow \text{NutProduct}(\text{peanutOil}) \\
&\rightarrow \text{NutAllergic}(\text{sebastian}) \\
&\rightarrow \exists \text{orderedDish.ThaiCurry}(\text{sebastian}) \\
&\rightarrow \text{Vegetarian}(\text{markus}) \\
&\rightarrow \exists \text{orderedDish.ThaiCurry}(\text{markus})
\end{aligned}
$$

Fig. 3. A simple example ELP rule base about food preferences. The variable v_s is assumed to be DL-safe.

- It allows role conjunctions for simple roles.
- It allows the use of DL-safe variable SWRL rules, in the sense that replacement of the safe variables by individuals in a rule must result in a valid $\mathcal{EL}^{++}$-rule.
- General DL-safe Datalog[16] rules are allowed.

The last point – allowing general DL-safe Datalog rules – is a bit tricky. As stated, it destroys polynomial complexity. However, if there is a global bound on the number of variables allowed in Datalog rules, then polynomiality is retained. Obviously, one would not want to enforce such a global bound; nevertheless the result indicates that a careful and limited use of DL-safe Datalog rules in conjunction with a polynomial description logic should not in general have a major impact on reasoning efficiency.

Since ELP is fundamentally based on $\mathcal{EL}^{++}$-rules, it features rules-style modelling in the sense in which $\mathcal{SROIQ}$-rules provide a rules modelling paradigm for $\mathcal{SROIQ}$. An example knowledge base can be found in Figure 3.

As for implementability, reasoning in ELP can be done by means of a polynomial-time reduction to Datalog, using standard Datalog reasoners. Note that TBox-reasoning can be emulated even if the Datalog reasoner has no native support for this type of reasoning. The corresponding algorithm is given in [12]. An implementation is currently under way.

Proposal: OWL 3 should feature a designated polynomial profile which is as large as possible, based on ELP.

A.6 Conclusions

Following the basic design principles for OWL, we made four suggestions for OWL 3.

[16] One could also simply allow DL-safe SWRL rules.

– OWL 3 should have a rule-based syntax based on Description Logic Rules.
– OWL 3 should allow the use of Boolean role constructors.
– OWL 3 should incorporate DL-safe SWRL and DL-safe variable SWRL.
– OWL 3 should feature a designated polynomial profile which is as large as possible, based on ELP.

We are aware that these are only first suggestions, and that a few open points remain to be addressed in research. We hope that our suggestions stimulate discussion which will in the end lead to a favorable balance between application needs and language development from first principles.