Suggestions for OWL 3*

Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton, Ohio

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.

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 [5] to OWL 2 [10] 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 paper, we present a number of suggestions for improvements to OWL DL,[1] 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

* This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) under the ReaSem project.

[1] 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 paper we focus on OWL DL.

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.

In Section 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 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 4, we suggest considerably extending OWL by including the DL-safe variable fragment of SWRL [2] together with the DL-safe fragment [8] of SWRL.

In Section 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 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.

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 [3].

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

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

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.

- 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.[2] 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,[3] 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

[2] The exact definition is a bit more complicated; see [3].
[3] 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

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 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 [7].

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

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 [9].

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 5.

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

4 DL-safe Variable SWRL

SWRL [2] 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* [8]. Syntactically, DL-safe rules are rules of the form

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

as in Section 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.[4] 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 [4] (see also Section 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 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

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

[4] 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.[5] 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.

5 Pushing The Tractable Profiles

The OWL 2 Profiles document [6] 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 [4] is a language with polynomial complexity which properly contains both OWL EL and OWL RL. It also contains most of OWL QL.[6] 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 2.[7] 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 $\mathcal{SROIQ}$.

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

$$C \sqcap \exists R.\{o\} \sqsubseteq \exists R_1.\text{Self}$$
$$D \sqcap \exists T.\{o\} \sqsubseteq \exists R_2.\text{Self} \quad \text{and}$$
$$R_1 \circ S \circ R_2 \sqsubseteq V.$$

See [3].

[6] Role inverses cannot be expressed in ELP.

[7] $\mathcal{EL}^{++}$-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[8] 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 [4]. An implementation is currently under way.

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

6 Conclusions

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

[8] 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 this paper stimulates discussion which will in the end lead to a favorable balance between application needs and language development from first principles.

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. 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/.
3. 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.
4. 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.
5. 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/.
6. 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/.
7. 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/.
8. B. Motik, U. Sattler, and R. Studer. Query-answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
9. 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.
10. 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/.