Modeling OWL with Rules: The ROWL Protégé Plugin

Md. Kamruzzaman Sarker[1], David Carral[1],
Adila A. Krisnadhi[1,2], and Pascal Hitzler[1]

[1] Wright State University, OH, USA
[2] Universitas Indonesia, Depok, Indonesia

Abstract. In our experience, some ontology users find it much easier to convey logical statements using rules rather than OWL (or description logic) axioms. Based on recent theoretical developments on transformations between rules and description logics, we develop ROWL, a Protégé plugin that allows users to enter OWL axioms by way of rules; the plugin then automatically converts these rules into OWL DL axioms if possible, and prompts the user in case such a conversion is not possible without weakening the semantics of the rule.

1 Motivation

It has long been argued, that rules are much more intuitive and easier to master than description logics, in terms of what their intended meaning is. We find this substantiated throughout our experiences as teachers and as ontology modelers which frequently work with domain experts.

To give just a simple example: The exact semantics behind a logical axiom such as

$$\mathsf{Journal} \sqsubseteq \forall \mathsf{publishedBy}.\mathsf{Organization}$$

in our experience often remains somewhat unclear even for people with significant exposure to ontologies and ontology modeling. On the other hand, a rule such as

$$\mathsf{Journal}(x) \wedge \mathsf{publishedBy}(x,y) \rightarrow \mathsf{Organization}(y)$$

is rather intuitive for most in its meaning, and can be both produced and processed much more readily.

The axiom and the rule just given are of course logically equivalent.[3] In fact many OWL axioms can be expressed equivalently as rules, which are, arguably, easier to understand and to produce.

As a consequence of these observations, we have produced a Protégé plugin which accepts rules as input, and adds them as OWL axioms to a given ontology, provided the rule is expressible by an equivalent set of such axioms. In case the rule is not readily transferable, the user is prompted and asked how to translate the rule, as there are different options how to do it in such cases. More information about the plugin can be found at `http://daselab.org/content/modeling-owl-rules`.

[3] When we interpret the rule in the sense of first-order predicate logic, i.e., according to the open world semantics.

2 Rules-to-OWL Transformation

In this section, we provide some examples of translations of rules into OWL axioms in an attempt to convey an intuitive understanding of our transformation. For a formal and complete of such procedure we refer the reader to [2]. Note that, as opposed to [2], we do not consider rules in our implementation that would require the use of role conjunction, as this is a logical constructor not currently allowed in OWL.

The following rule can be used to characterize all individuals taking courses and working for a department as student workers.

$$\mathsf{attends}(x,y) \wedge \mathsf{Course}(y) \wedge \mathsf{worksFor}(x,z) \wedge \mathsf{Dept}(z) \rightarrow \mathsf{StudentWorker}(x) \quad (1)$$

We transform this rule into a DL axiom via a series of equivalence preserving transformations. First, we detect that both y and z are variables that can be "rolled up," as they only occur in a single object property. Thus, these variables can be sequentially removed from the rule, resulting in the following:

$$\exists \mathsf{attends}.\mathsf{Course}(x) \wedge \mathsf{worksFor}(x,z) \wedge \mathsf{Dept}(z) \rightarrow \mathsf{StudentWorker}(x)$$
$$\exists \mathsf{attends}.\mathsf{Course}(x) \wedge \exists \mathsf{worksFor}.\mathsf{Dept}(x) \rightarrow \mathsf{StudentWorker}(x)$$

Furthermore, we can unify all unary atoms of the form $C(x)$, i.e., sharing the same variable x, yielding:

$$(\exists \mathsf{attends}.\mathsf{Course} \sqcap \exists \mathsf{worksFor}.\mathsf{Dept})(x) \rightarrow \mathsf{StudentWorker}(x)$$

The previous rule can then be directly translated into OWL as the following axiom:

$$\exists \mathsf{attends}.\mathsf{Course} \sqcap \exists \mathsf{worksFor}.\mathsf{Dept} \sqsubseteq \mathsf{StudentWorker}$$

For the next example, we have the following rule, which specifies that "all mice are smaller than all elephants."

$$\mathsf{Mouse}(x) \wedge \mathsf{Elephant}(y) \rightarrow \mathsf{smallerThan}(x,y)$$

Translating such a rule into OWL requires us to first connect the variables in the body. We do so by adding atoms of the form $U(t,u)$ with U the universal property, i.e., owl:topObjectProperty.

$$\mathsf{Mouse}(x) \wedge U(x,y) \wedge \mathsf{Elephant}(y) \rightarrow \mathsf{smallerThan}(x,y)$$

The previous rule can be directly translated into OWL resulting in the following three axioms where R_{Mouse} and R_{Elephant} are fresh object properties not previously occurring in the ontology:

$$\mathsf{Mouse} \sqsubseteq \exists R_{\mathsf{Mouse}}.\mathsf{Self}$$
$$\mathsf{Elephant} \sqsubseteq \exists R_{\mathsf{Elephant}}.\mathsf{Self}$$
$$R_{\mathsf{Mouse}} \circ U \circ R_{\mathsf{Elephant}} \sqsubseteq \mathsf{smallerThan}$$

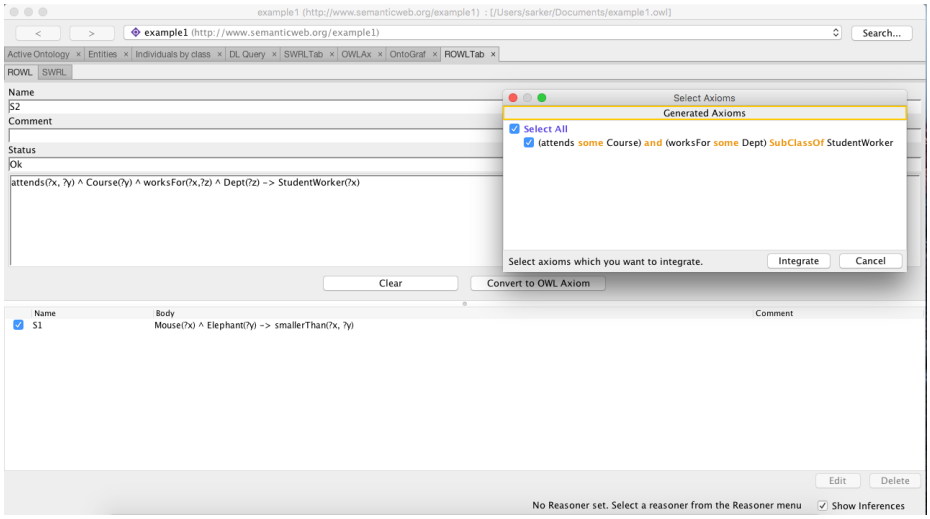

Fig. 1. The ROWL interface. The pop-up window appears after clicking "Convert to OWL Axiom" button and the transformation is successful.

Certain rules cannot be expressed in OWL employing our approach. For example, the following rule, which characterizes the set of individuals taught by their own uncle, cannot be translated by our approach.

$$\mathsf{hasFather}(x,y) \wedge \mathsf{hasBrother}(y,z) \wedge \mathsf{taughtBy}(x,z) \rightarrow \mathsf{TaughtByUncle}(x)$$

Note that, such a rule cannot be reduced in the same way as rule , since every variable occurs in at least two atoms with object properties as predicates.

In cases, such as the previous one, in which a rule cannot be translated into OWL using a set of DL axioms, our implementation will suggest several options to translate such rule using nominal schemas [1]. The chosen option by the user will be recorded in an annotation which will be added to the rule. As of right now, there is no syntax for nominal schemas in OWL and thus, we have decided that an annotation is the best way to convey such information.

3 Plugin Description and Features

Figure 1 depicts the user interface of the ROWL plugin. This plugin is implemented on top of Protégé's SWRLTab plugin implementation and thus, it borrows the pretty much SWRLTab user interface for entering rules as input. As seen in the figure, the plugin consists of two tabs: ROWL and SWRL. The latter is really SWRLTab input interface, while the former is a modification of the latter where we add "Convert to OWL Axioms" button. A user can enter a rule in ROWL tab using the standard SWRL syntax, e.g.:

$$\mathsf{attends}(?x, ?y) \text{ ^ } \mathsf{Course}(?y) \text{ ^ } \mathsf{worksFor}(?x, ?z) \text{ ^ } \mathsf{Dept}(?z) \text{ -> } \mathsf{StudentWorker}(?x)$$

When the "Convert to OWL Axiom" button is clicked, ROWL will attempt to apply the rules-to-OWL transformation described in the previous section to the given rule. If successful, a pop-up will appear displaying one or more OWL axioms resulted from the transformation, presented in Manchester syntax. These axioms can then be integrated into the active ontology.

If the given rule cannot be transformed into OWL axiom, ROWL will prompt the user if they still want to insert the rule into the ontology as an SWRL rule with annotation. If the user agrees, ROWL will switch to its SWRL tab and proceeds in the same way as adding a rule via the original SWRLTab. Note that ROWL is separate from the original SWRLTab, hence any SWRL rule added via ROWL will not affect rules added through the original SWRLTab.

Note that once the axioms generated from rules are added into the ontology, the plugin does not provide a way to undo it and recover the original rule from which the axioms were generated. That is, when a user enters a rule through this plugin and converts it to OWL axioms, the active ontology is either augmented with the generated OWL axioms or SWRL rules. To have this feature, we need a way to record which axioms were generated from which rules. This will be considered as part of future development of this plugin.

Finally, a feature of ROWL not found in SWRLTab is the possibility to automatically add declarations for classes and properties if the inserted rule contain classes or properties not yet defined in the ontology. For example, in the rule above, the original SWRLTab requires that attends and worksFor to be already defined as object property, while Course, Dept, and StudentWorker as class in the ontology. This would add a little bit more freedom for the user to enter any rule (s)he wishes during modeling because (s)he does not need to first exit the plugin and declare the classes and properties directly in Protégé.

Acknowledgements. This work was supported by the National Science Foundation under award 1017225 *III: Small: TROn – Tractable Reasoning with Ontologies*.

References

1. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for OWL: nominal schemas for integrating rules and ontologies. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 – April 1, 2011. pp. 645–654. ACM (2011)
2. Martínez, D.C., Hitzler, P.: Extending description logic rules. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V. (eds.) The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7295, pp. 345–359. Springer (2012), http://dx.doi.org/10.1007/978-3-642-30284-8_30