OWL to ASP Translator

Anirudh Acharya
School of Computing, Informatics, and Decision Systems, ASU
aachary8@asu.edu

Abstract

This project is about translating OWL ontologies to an Answer Set Programming Language like f2lp, so that reasoning can be done on the ontology. In this project we investigate the possible uses of representing an OWL ontology as an ASP program and, query this knowledge base and compare it to other OWL reasoners like FaCT.

Introduction

Semantic web initiative is a plan to revolutionize the World Wide Web by representing Web content in more machine-processable form. This will make tasks such as information retrieval and information integration much easier and more efficient. The Web Ontology Language is one such representation that aims to make Web content more machine understandable. OWL is a family of knowledge representation or ontology languages to represent knowledge bases. OWL was built as an extra layer over another knowledge representation language, RDF (Resource Description Framework). Though RDF served a wide variety of purposes it had certain features missing such as, expressing disjointness of classes, cardinality restrictions, and capability to define local scope of object properties. OWL was created as an ontology language to overcome these shortcomings of RDF. OWL has been very wisely applied to tasks such as information integration and metadata management. There are also several reasoners such as Pellet, and FaCT+, which have been built to reason over these ontologies.

OWL is actually a family of three sublanguages – OWL-Lite, OWL-DL, and OWL-Full, each with different levels of expressiveness. Every OWL sublanguage is upward compatible, meaning every OWL-Lite ontology is a legal OWL-DL ontology, every OWL-DL ontology is a legal OWL-Full ontology.

OWL-Full uses all the OWL language primitives and is fully upward compatible, semantically and syntactically, with RDF and RDFS. OWL-Full is expressive is expressive to the extent of being undecidable, hence the reasoning support is less efficient. OWL-DL is a sublanguage of OWL-Full, is computationally complete and decidable. It has all the OWL constructs but they can be used only under certain restrictions, such as the cardinality rule can only be used with object property. OWL-DL is also not fully compatible with RDF. OWL-Lite is a further restriction of expressivity compared to OWL-DL. It does not support disjoint statements and enumerated classes. This high restriction of expressivity makes reasoning over OWL-Lite ontologies very simple and easy.

In this project we deal with OWL ontologies built with OWL-DL language, based on Description Logics. Despite the wide range of expressivity of OWL language, it does have certain shortcomings, such as, it does not allow closed-world reasoning, and introducing integrity constraints and finding possible Herbrand models given the constraints in the ontology.

Adding a layer of Answer Set Program over an OWL ontology or representing the OWL ontology as an Answer

Set Program can fix these shortcomings of OWL language. In this project we tried to address this issue by investigating these use cases which would be hard to handle with OWL-DL alone.

By trying to combine OWL-DL semantics with Answer Set Programming, we tried to come up with a knowledge representation model that is compatible with both logic programs and OWL-DL semantics, and we try to address some of the situations, where OWL reasoners fail, as mentioned earlier.

OWL – Description Logic

OWL-DL is based on Description Logics (DL), a knowledge representation method based on first-order logic.

An OWL-DL knowledge base consists of - owl classes, representing sets of objects, object or data property representing the different roles of the owl classes and the relationships between them, and individuals, representing specific objects.

Using a rich set of owl classes and object properties, we can construct complex concepts, which describe the conditions for concept membership. For example, the concept ∃hasBase.PizzaBase describes those objects that are related through the hasBase role with an object from the concept PizzaBase.

An OWL-DL knowledge base KB typically consists of a TBox T and an ABox A. A TBox contains axioms about the general structure of all owl classes, and is therefore lays the constraints as to what sort of relationships different owl classes can have amongst themselves. For example, the TBox axiom

Pizza ⊑ ∃hasBase .PizzaBase

States that every instance of class Pizza will be related to some instance of class PizzaBase by the role hasBase

An ABox contains axioms about the structure and relationships of particular instances of different owl classes. For example, the below DL rules mean that,

Person(Peter)
hasBrother (Peter ,Paul)

The individual Peter belongs to the owl class Person. The second rule says that the individuals Peter and Paul are related by the relationship hasBrother.

Answer Set Programming

Logic programming is a Knowledge Representation system where a particular domain is described using logical rules and facts. A logical rule is of the form:

$H \leftarrow B_1, B_2 \ldots, B_n, \text{not } C_1, C_2 \ldots C_n$

Where H is the head of the rule and the RHS of the equation is the body of the rule.

A set of such rules R can have any number of stable models, and such rules with a combination of disjunctions in the rule heads and extensions with classical negation is called an Answer Set Programming.

Answer Set Program typically deals with query answering over a finite data set, and it is often used in data-intensive applications. Answer Set programming is very useful for representing computationally hard problems.

Translating DL Rules to Answer Set Program

The logic rules in an Answer Set Program follow First Order Logic semantics and OWL-DL ontologies can be represented using Description Logic rules. So the task of translating an OWL ontology into an ASP program is that of translating DL rules into their corresponding first order logic rules.

The logic rules in an ASP program require to be grounded. The grounding in this translation is provided by the individuals of the OWL ontology, so the reasoning over the OWL ontology happens over the individuals of the ontology.

Also each owl class is represented as a unary predicate. For instance, if we have to denote that PepperoniPizza belongs to class Pizza then we write

$$
\begin{aligned}
t(\top, x) &\mapsto \texttt{true} \\
t(\bot, x) &\mapsto \texttt{false} \\
t(A, x) &\mapsto A(x) \\
t(C_1 \sqcap C_2, x) &\mapsto t(C_1, x) \wedge t(C_2, x) \\
t(C_1 \sqcup C_2, x) &\mapsto t(C_1, x) \vee t(C_2, x)
\end{aligned}
$$

$$
\begin{aligned}
t(\neg C, x) &\mapsto \neg t(C, x) \\
t(\exists R.C, x) &\mapsto \exists y.R(x, y) \wedge t(C, y) \\
t(\forall R.C, x) &\mapsto \forall y.R(x, y) \Rightarrow t(C, y) \\
t(C \sqsubseteq D) &\mapsto \forall x.t(C, x) \Rightarrow t(D, x) \\
t(a{:}C) &\mapsto t(C, a) \\
t((a, b){:}R) &\mapsto R(a, b) \\
\geqslant nP &\mapsto \left| \exists y_1 \ldots y_n . \bigwedge_{k=1}^{n} P(x, y_k) \wedge \bigwedge_{i<j} y_j \neq y_j \right| \\
\leqslant nP &\mapsto \left| \forall y_1 \ldots y_{n+1} . \bigwedge_{k=1}^{n+1} P(x, y_k) \supset \bigvee_{i<j} y_i = y_j \right|
\end{aligned}
$$

- Example:

$$
\begin{aligned}
&t(HappyFather \sqsubseteq Man \sqcap \exists hasChild.Female) = \\
&\quad \forall x.HappyFather(x) \Rightarrow (Man(x) \wedge (\exists y.hasChild(x, y) \wedge Female(y)))
\end{aligned}
$$

Figure 1: Conversion from DL Rules to First Order Logic

Pizza(PepperoniPizza).

Every Object Property is represented as a binary predicate. For example, if we have to represent that Pepperoni Pizza has a Peperoni Topping then we write

hasTopping(PepperoniPizza, PepperoniTopping).

Every owl class and object property is represented in a choice rule, ranging over the individuals as the grounding values.

We then translate the various relationships between classes and object properties as constraints. The models obtained from such a program will be the models permissible by the given OWL-DL knowledge base. See Figure 1 for a more detailed list of DL rules to FOL conversion.

The cardinality rule is also expressed using choice formulas. For example to show that a particular pizza can have only 1 pizza base we write the rule as

1{hasBase(Z,Z1):pizzaBase(Z1)}1 <- pizza(Z).

A sample translation for a mini pizza ontology and wine ontology is demonstrated in the Appendix.

Results

A translated ASP program can be useful for querying the knowledge base in two main ways –

1. Check if a particular rule or condition is entailed by the ontology, integrity constraints.
2. Query for individuals that meet a certain criteria, find models that satisfy the constraints in the ontology.

When tested with the mini pizza ontology, the following query
![Z]: (queryPizza(Z) -> pizza(Z) & ?[A]: hasTopping(Z,A) & vegetableToping(A)).

Returns the models
queryPizza(tuePizza), queryPizza(thurPizza)
We could display other predicates from the program and see what are the models that satisfy the constraints imposed by the OWL ontology.

We also check for integrity constraints by adding the following additional constraint –

?[Z]: (pizza(Z) -> hasTopping(Z,Z1) & hasTopping(Z,Z2) & vegetableTopping(Z1) & meatTopping(Z2)).

The resulting program will return zero models, telling that in the ontology it is not possible for a pizza to have both a vegetable and a meaty topping.

Conclusion

In this project we did a study of OWL-DL ontology and syntax, came up with a translation for DL rules into First order logic. We also investigated the possible shortcomings of OWL reasoners like FaCT, and tried to address it by

demonstrating how they can be overcome by using Answer Set Programming. We presented the features of representation with non-trivial translations and query expressions. We also gained insight into how OWL could be integrated with ASP rules without sacrificing semantic compatibility with either representations.

The project could be extended to incorporate more properties like functional properties, inverse functional properties, transitive property, etc… Also currently certain cardinality rules do not get translated, we would like to add these features going ahead and see how much better the system can be made. A query system with a combination of OWL representation and ASP reasoning can be a very promising and interesting line of study.

References

Motik Boris, Horrocks Ian, Rosati Riccardo, and Sattler Ulrike, Can OWL and Logic Programming Live Together Happily Ever After?

Antoniou Grigoris, and Harmelen Frank van, A Semantic Web Primer, The MIT Press, Cambridge Massachussets, London, England.