`Wolpertinger`: A Fixed-Domain Reasoner

Sebastian Rudolph, Lukas Schweizer, and Satyadharma Tirtarasa

Computational Logic Group, TU Dresden, Dresden, Germany
`firstname.lastname@tu-dresden.de`

1 Introduction

The Web Ontology Language OWL [5] comes with comprehensive modeling tool support. This sometimes leads to situations where OWL is the modeling paradigm chosen over other formalisms, even if the application scenario does not match the typical usage of this language. For example, problems of a constraint-satisfaction nature do not go well with OWL's standard semantics.

Consider a toy configuration problem: Five people want to go home after a party in one car having five seats. The situation is conveniently (and seemingly adequately) modeled using OWL (for brevity, we use DL-style notation):

five persons their gender and intoxication:
$Woman \sqcap Drunk(alice)$, $Woman(claire)$, $Woman(eve)$,
$Man \sqcap Drunk(bob)$, $Man \sqcap Drunk(dan)$,
five seats, where the driver's seat must be occupied by a sober person:
$Seat \sqcap \exists sits^{-}.Person \sqcap \neg Drunk(fl)$, $Seat(fr)$, $Seat(bl)$, $Seat(bm)$, $Seat(br)$,
relative location of the seats (and symmetry of seat-neighbourhood)
$next(fr, fl)$, $next(br, bm)$, $next(bm, bl)$, $next \sqsubseteq next^{-}$,
some elementary background knowledge
$Woman \sqcup Man \sqsubseteq Person$, $Person \sqsubseteq \neg Seat$,
everyone has to be seated (in distinct seats), women don't want to sit next to men:
$Person \sqsubseteq \exists sits.Seat$, $\top \sqsubseteq \leqslant 1 sits^{-}.\top$, $Woman \sqsubseteq \forall sits^{-}.\forall next.\forall sits.\neg Man.$

It is easy to see there is no seating satisfying the given constraints. However, an OWL reasoner will still (rightfully) indicate that the given ontology is satisfiable, since the open-world semantics of OWL allows for the existence of anonymous individuals (in our example, a model might contain more than the indicated persons and seats).

To overcome these problems and to enable the usage of OWL in those settings, we proposed *fixed-domain reasoning* under a non-standard model-theoretic semantics that restricts the domain to an explicitly given finite set (in our example, this would be the 10 explicitly mentioned entities). We also showed that for expressive ontology languages, this semantics has a significantly lower complexity than the standard semantics [2, 4].

We observed that the fixed-domain semantics can be axiomatized in OWL, allowing to employ available OWL reasoners for standard reasoning tasks. Still, we showed that available OWL reasoners struggle with standard reasoning under this new semantics [2]. Besides standard reasoning tasks, a particular requirement arising from industrial collaboration is the need to *enumerate models*. Note that in our considered toy scenario, it might not just be interesting *if* a seating is possible, but – in the positive case – also, *what* these seatings are.

2 Implementation

To implement efficient fixed-domain reasoning in OWL, we proposed a translation of arbitrary OWL DL ontologies into *answer set programming* (ASP) [1]. This allows us to leverage existing optimized ASP machinery to perform both standard reasoning as well as the non-standard tasks *model enumeration* and query entailment quite elegantly.

The translation takes an OWL DL ontology $\mathcal{O}$ and a given fixed domain Δ and produces a corresponding answer-set program $\Pi(\mathcal{O}, \Delta)$ whose answer sets correspond to the models of $\mathcal{O}$ with domain Δ. Intuitively, the set of all fixed-domain interpretations defines a search space, which can be traversed searching for models, guided by appropriate constraints. These constraints correspond exactly to the axioms of the given ontology (after a structural transformation has been carried out). Thereby interpretations violating any of the given axioms are ruled out as models. For more details, we refer the interested reader to our previous work [2].

2.1 Implementation Overview

The mentioned translation was implemented, and is available in the open-source tool `Wolpertinger`.[1] Figure 1 abstractly depicts the main components of the tool and their interplay. At the core, we provide an implementation of the `OWLReasoner` interface, which is accessible either via a command-line interface, or from within Protégé for which we additionally provide a reasoner plugin. In the following, we describe all features and the realizing components.

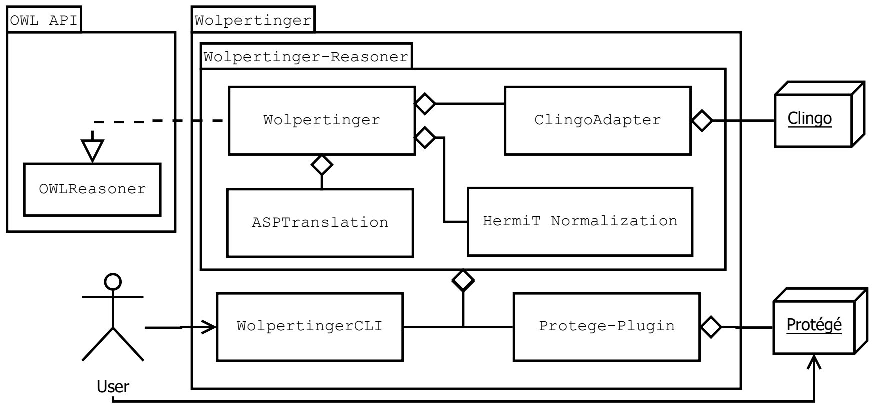

Fig. 1. Abstract Component Overview.

[1] https://github.com/wolpertinger-reasoner/Wolpertinger

2.2 Components and Features

Wolpertinger This component realizes the OWL API's `OWLReasoner` interface, for which many of the defined reasoning tasks are supported.[2] Moreover, it orchestrates the translation process, consisting of (1) normalizing the input ontology, (2) translating the normalized ontology into an answer-set program, and finally (3) managing the communication with the ASP solver `clingo` [3], which computes as many answer sets as requested each representing a model of the input ontology. The translation and interaction is encapsulated, thus the user is not aware of the internal mechanics. Model enumeration is a functionality beyond what is defined in `OWLReasoner` and is therefore one of the functionalities provided outside that interface. User interfaces to access `Wolpertinger` are (1) a command line interface, and (2) a Protégé reasoner plugin.

WolpertingerCLI The command line interface is currently the only user interface that gives access to all supported features. In short, the supported functionalities are:

Model Enumeration:	Generate 1, n or all models.
Consistency Checking:	Check if there is any model with the given domain for the input.
Entailment Checking:	Given the input ontology $\mathcal{O}$ and another ontology $\mathcal{O}'$, check if $\mathcal{O}'$ is entailed by $\mathcal{O}$.
Classification:	Output all subclasses/superclasses of a given class C.
Justification:	In case of inconsistency, provide a minimal set of axioms that cause the inconsistency.
Instance Retrieval:	Provide all individuals that are an instance of a given class.
Individual Types:	Provide all classes of which a given individual is an instance.

Regarding the specification of the fixed domain, two options are supported: either the domain is implicitly built by collecting the individuals occurring in the given input ontology, or the explicit domain is extracted from the named individuals in another given OWL file.

Protégé-Plugin We provide a preliminary implementation of a reasoner Protégé-plugin, such that `Wolpertinger` can conveniently be used in the known way (like other available OWL reasoners) from within Protégé. At the moment, the plugin only gives access to the standard reasoning tasks, under the fixed-domain semantics. Especially the enumeration feature has not yet been made available from within Protégé and remains as future work.

3 Conclusion and Future Work

We presented a tool for reasoning in OWL under the fixed-domain semantics, where models are confined to a given form. This is particularly adequate for OWL ontologies representing constraint-type problems. Although the usage of OWL (as opposed to,

[2] At the current stage, we support all of OWL DL except data properties.

say, full first-order logic) still imposes some restrictions regarding expressivity (e.g., by restricting the predicate arity to 1 and 2), quite large and involved problem scenarios can be modeled by OWL ontologies. In fact, `Wolpertinger` is being used successfully for solving configuration problems in industry projects.

Clearly, more comprehensive evaluations of our system with respect to such ontologies remain as imperative issue. Moreover, ASP is only one possible target formalism for translating fixed-domain reasoning problems; other formalisms are also conceivable, including pure CSP languages or even SAT. An implementation and comparative evaluation of these alternatives will allow us to pick the most efficient formalism.

In the future, we also plan to extend the presented work in several directions. We will provide a Protégé-plugin for the added functionality beyond the standard reasoning tasks, such as the enumeration of models and functions to handle models (i.e. exporting them to proper OWL formats). We plan to incorporate typical ontology engineering tasks such as explanation and axiom pinpointing into our framework, for which we have preliminary results already. Finally, we will consider extensions of the source formalism OWL, e.g. by non-monotonic features. As ASP itself is a non-monotonic logic programming formalism, rule-based extensions of OWL – monotonic or non-monotonic – should be straightforward to accommodate.

References

1. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM 54(12), 92–103 (2011)
2. Gaggl, S.A., Rudolph, S., Schweizer, L.: Fixed-Domain Reasoning for Description Logics. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 819 – 827. IOS Press (September 2016)
3. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: *Clingo* = ASP + control: Preliminary report. In: Leuschel, M., Schrijvers, T. (eds.) Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14). vol. arXiv:1405.3694v1 (2014), theory and Practice of Logic Programming, Online Supplement
4. Rudolph, S., Schweizer, L.: Not too big, not too small... complexities of fixed-domain reasoning in first-order and description logics. In: Oliveira, E., Gama, J., Vale, Z.A., Cardoso, H.L. (eds.) Progress in Artificial Intelligence - 18th EPIA Conference on Artificial Intelligence, EPIA 2017, Porto, Portugal, September 5-8, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10423, pp. 695–708. Springer (2017), https://doi.org/10.1007/978-3-319-65340-2_57
5. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (2009)