

# FOWL – An OWL to FOL Translator

Simon Flügel<sup>1</sup>, Anna Kleinau<sup>1</sup>, Fabian Neuhaus<sup>1,2</sup>, Martin Glauer<sup>1</sup> and Janna Hastings<sup>1</sup>

<sup>1</sup>Otto von Guericke University Magdeburg, Germany

<sup>2</sup>Free University of Bozen-Bolzano, Bozen, Italy

## Abstract

The Web Ontology Language OWL 2 DL is the language used to build ontologies for a wide range of domains including the life sciences, the financial industry and the domain of energy simulations. Many of these domain ontologies are built beneath upper ontologies (such as DOLCE or BFO), which themselves are written in more expressive logics than OWL 2 DL, in particular first-order logic (although they may provide truncated versions in OWL). In this paper we present FOWL, a tool that enables the translation of OWL ontologies into FOL, and thereby the integration of OWL domain ontologies with first-order logic ontologies. One use case for this tool is the ability to validate OWL domain ontologies against their upper-level ontology. FOWL also allows reasoning over the translation using the VAMPIRE reasoner.

## Keywords

Web Ontology Language, OWL, first-order logic, ontology

## 1. Introduction

In the applied ontology community the two most popular ontology languages are the Web Ontology Language (OWL) [1], more precisely OWL 2 DL and more restrictive OWL 2 profiles, and first-order logic (FOL). Both languages have their advantages and disadvantages – FOL is more expressive than OWL 2 DL, but the satisfiability problem for FOL ontologies is undecidable. In contrast, the direct semantics for OWL 2 DL [2] is based on the description logic SHROIQ [3] and, thus, it is decidable whether an OWL 2 ontology is satisfiable. Further, the OWL Manchester syntax [4] for OWL 2 DL is relatively easy to learn for domain experts.

Often, FOL is used for upper level ontologies, where additional expressivity is needed. Examples of such are DOLCE [5] and BFO [6]. This leads to the unfortunate situation that domain ontologies, which are supposed to be based on upper level ontologies, are written in OWL, while the canonical versions of these upper level ontologies are written in FOL. Therefore, the consistency of the domain ontologies with respect to the FOL-versions of their associated upper level ontologies cannot be verified. For this a tool is needed that supports the integration of OWL ontologies with FOL ontologies.


Another use case for the integration of OWL and FOL ontologies is the extension of a OWL ontology with FOL axioms. For example, in [7] an OWL ontology was complemented by a Common Logic ontology (a variant of FOL) to supplement the OWL ontology with knowledge

---

FOIS 2021 Demonstrations, held at FOIS 2021 - 12th International Conference on Formal Ontology in Information Systems, September 13-17, 2021, Bolzano, Italy



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

that cannot be expressed in OWL. However, without tool support it is not possible to integrate both into a whole that can be reasoned with.

In this paper we present FOWL, a tool that enables the automatic translation of OWL ontologies into FOL (in particular, the TPTP syntax for first-order theories). We start by describing our approach to the translation on a theoretical level. Afterwards we take a look at how FOWL works and can be used in practice. We also evaluate the performance of FOWL and discuss related work as well as future tasks.

## 2. Approach

Our tool translates OWL 2 DL to TPTP. The TPTP syntax [8] has been selected because of its widespread use among FOL reasoners. This gives us the opportunity to connect FOWL directly to reasoners such as VAMPIRE [9] or E [10]. OWL 2 has two different semantics: The RDF-based Semantics [11] and the Direct Semantics [2]. The RDF-based Semantics is an extension of the semantics of RDFS with datatypes. The Direct Semantics provides a model theory for OWL 2 DL, which is compatible with the description logic SROIQ [3]. Since SROIQ is a decidable fragment of FOL, a translation from OWL 2 to FOL based on the Direct Semantics is more straightforward than using the RDF-based Semantics. For this reason, FOWL's translation is based on the Direct Semantics.

The OWL 2 Direct Semantics distinguishes between the object domain and the (disjoint) data domain. The object domain is the extension of *owl:Thing*; individuals are elements and classes are subsets of the object domain. The data domain is the extension of *rdfs:Literal*; literals are elements and datatypes are subsets of the data domain. Thus, one natural way to translate OWL 2 DL to FOL is to use a typed first-order logic, and to treat *owl:Thing* and *rdfs:Literal* as different types. An alternative approach is to use untyped FOL, to treat *owl:Thing* and *rdfs:Literal* as unary predicates and to introduce axioms that capture the information that would otherwise be represented by the type system. TPTP provides formats for both typed and untyped FOL. Since the untyped fof-format is supported by a wider range of FOL reasoners, we decided to use the second approach and translate OWL 2 formulae into the untyped fof-TPTP.

## 3. The Translation of OWL to FOL

For our translation, we have defined a mapping  $[ \ ]_q^p$  which maps OWL 2 DL axioms and OWL expressions that are part of axioms to sets of FOL expressions. It has two parameters  $p, q$ , which are used to keep track of substitutions that need to be made. When the translation function is called on an axiom, the parameters are empty, i.e.  $p = q = \emptyset$ . If a parameter is empty, we omit it and, for example, write  $[E]^p$  instead of  $[E]_{\emptyset}^p$ .

With this mapping, we have determined a translation for each type of OWL expression. The mapping can then be applied recursively in order to translate an OWL axiom that consists of several OWL expressions into a corresponding FOL expression or a set of such. E.g., OWL axioms of the form *SubClassOf*( $A, B$ ) are translated with a FOL sentence  $\forall x([A]^x \rightarrow [B]^x)$ . In natural language this means, that, if  $A$  is a subclass of  $B$ , then every individual which belongs to class  $A$  also belongs to class  $B$ . Note that  $A$  and  $B$  themselves are not yet translated, but

are OWL class expressions and we need to apply our translation function again using the parameter  $x$  wherever a variable is required. The complete translation function can be found on the project’s wiki page<sup>1</sup>. For the sake of brevity we just illustrate it in this paper with a short example:

$$\begin{aligned}
& [\text{SubClassOf}(\text{Fish}, \text{ObjectIntersectionOf}(\text{Animal}, \text{SomeValuesFrom}(\text{hasPart}, \text{Gills})))] \\
& \Leftrightarrow \forall x([\text{Fish}]^x \rightarrow [\text{ObjectIntersectionOf}(\text{Animal}, \text{SomeValuesFrom}(\text{hasPart}, \text{Gills}))]^x) \\
& \Leftrightarrow \forall x([\text{Fish}]^x \rightarrow ([\text{Animal}]^x \wedge [\text{SomeValuesFrom}(\text{hasPart}, \text{Gills})]^x)) \\
& \Leftrightarrow \forall x([\text{Fish}]^x \rightarrow ([\text{Animal}]^x \wedge \exists y([\text{hasPart}]_y^x \wedge [\text{Gills}]^y)) \\
& \Leftrightarrow \forall x(\text{Fish}(x) \rightarrow (\text{Animal}(x) \wedge \exists y(\text{hasPart}(x, y) \wedge \text{Gills}(y))))
\end{aligned}$$

As this example illustrates, the translation function is the result of a straightforward translation of the Direct Semantics into FOL. In this example we use both substitution parameters, since a second variable is needed to translate the existential restriction. For the purpose of translating OWL to FOL it is sufficient to keep track of only two variables as parameters (as our translation function illustrates). This might be counter-intuitive, since SROIQ contains features beyond the two-variable fragment of FOL like, for example, transitivity axioms and counting quantifiers. However, the translation of these axioms only requires to use additional new variables, and does not require the reuse of more than two variables that have been introduced before. Therefore, two substitution parameters suffice.

An OWL ontology is translated to FOL by translating all of its axioms and adding some background axioms. Some of them axiomatise the distinction between objects and data in OWL and provide a kind of ‘soft-typing’ as discussed in Section 2. For example, for any class  $C$  in the signature of the OWL ontology, a background axiom  $\forall x(C(x) \rightarrow owl:Thing(x))$  is added to the FOL translation. Other background axioms capture assumptions of the Direct Semantics (e.g., the extension of *owl:Thing* and *rdfs:Literal* are nonempty). A third kind of background axiom provides the axiomatisation for the OWL reserved vocabulary (e.g.,  $\forall x \neg owl:Nothing(x)$ ).

## 4. FOWL – Architecture and Implementation

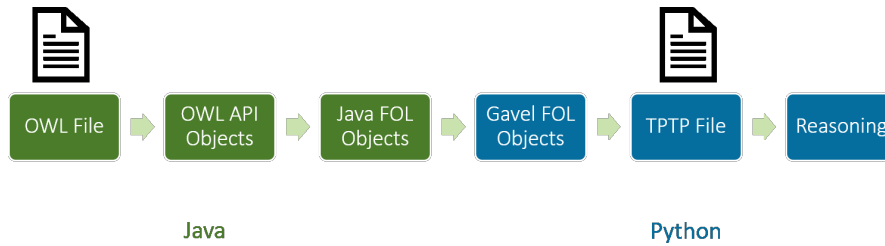
The structure of our implementation as shown in Figure 1 is based on the two tools we used, namely the *OWL API* and *Gavel*. The *OWL API* is a Java toolkit to create, load and change OWL ontologies. It represents the ontology as objects that can be traversed when working with the ontology. It also provides access to OWL reasoners such as Hermit [12]. *Gavel*<sup>2</sup> is a Python tool to create and work with FOL files. It provides functions to create new FOL axioms and objects and methods to export them in TPTP or string format. It provides access to FOL reasoners such as VAMPIRE [9].

The goal of our tool is to translate OWL ontologies into first order logic. This means both the OWL tools of the *OWL API* and the FOL tools of *Gavel* provide important functionality for FOWL. Because the *OWL API* is only available in Java and *Gavel* only in Python, FOWL uses both languages. FOWL is started and stopped from Python. The Python script then starts a Java server that handles the OWL ontology and communicates with Python via Py4J<sup>3</sup>.

<sup>1</sup><https://github.com/gavel-tool/python-gavel-owl/wiki/OWL-to-FOL-mapping>

<sup>2</sup><https://github.com/gavel-tool/python-gavel>

<sup>3</sup><https://www.py4j.org/>



**Figure 1:** The FOWL pipeline

When the Python script starts the Java server, it specifies as a parameter the ontology which is to be translated. This can be a local file or an IRI. The ontology then gets loaded by the OWL API, along with any imported ontologies. The visitor pattern, an easy and efficient way to visit each object of the ontology, is used to traverse recursively through the ontology and create the FOL translation. The translation is then saved in classes that mirror Gavel’s FOL classes and can be identified by their name function. These classes are provided to the Python script, which iterates through the sentences provided by the Java server and translates them in a 1:1 style to the appropriate Gavel classes. For reasoning over the translation, the sentences are then parsed into the TPTP format.

By providing a text file with conjectures in TPTP format, the VAMPIRE reasoner can then reason over those conjectures. Gavel provides access to the reasoning result, as well as the steps leading to it. The proof status is classified by using the SZS ontologies <sup>4</sup>.

FOWL can be accessed via a command line interface. It features functions that translate ontologies and provide access to OWL reasoner Hermit as well as FOL reasoners VAMPIRE and E. More detailed instructions on how to install and use FOWL can be found on the project’s Github page <sup>5</sup>. FOWL can be used according to the GNU Affero General Public License.

## 5. Evaluation

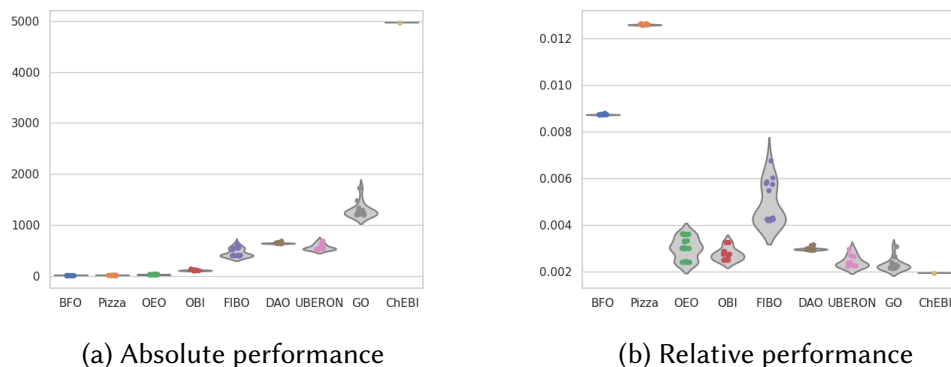
**Performance** We evaluated the performance of FOWL for translating different OWL ontologies to TPTP. The ontologies are Basic Formal Ontology (BFO) [6], Pizza Ontology <sup>6</sup>, Open Energy Ontology (OEO) [13], Ontology for Biomedical Investigations (OBI) [14], Financial Industries Business Ontology [15], Drosophila Anatomy Ontology (DAO) [16], Uberon [17], Gene Ontology (GO) [18, 19] and Chemical Entities of Biological Interest (ChEBI) [20].

The results are shown in Figure 2. BFO and the Pizza Ontology, which are the smallest ontologies in our evaluation, have a significantly higher time-per-axiom-ratio than the larger ontologies. This is possibly a result of the overhead due to background axioms and function calls which is unaffected by the ontology size. Figure 2b also shows that strongly axiomatized ontologies such as OBI do not lead to a significant performance loss compared to weakly

<sup>4</sup><http://www.tptp.org/Seminars/SZSOntologies/Summary.html>

<sup>5</sup><https://github.com/gavel-tool/python-gavel-owl>

<sup>6</sup><https://protege.stanford.edu/ontologies/pizza/pizza.owl>



**Figure 2:** Translation times for different ontologies. 2a shows the seconds taken for the complete translation, 2b the seconds per axiom. The ontologies are sorted from smallest to largest.

axiomatized ontologies. This implies that the type of axioms does not have a big influence on the translation performance.

**Consistency** If FOWL’s translation is correct, it should preserve the (in)consistency of an ontology across the translation. Thus, in order to evaluate the quality of the translation of FOWL, we ran the following experiment: We created a set of consistent and a set of inconsistent OWL ontologies, translated them with FOWL to FOL, and evaluated the consistency of the result using VAMPIRE.

For this purpose we extracted modules of ChEBI based on randomly chosen classes using ROBOT<sup>7</sup>. Since ChEBI is consistent, all of its modules are consistent. In our experiment we considered 168 consistent ontologies. To create inconsistent ontologies, we used modules of ChEBI as starting point, but added additional axioms that lead to an inconsistency. For this purpose an OWL reasoner was utilised to classify a given ChEBI module, and a random theorem was selected for the purpose of generating a contradiction. E.g., if for a given ChEBI module a theorem  $Subclass(A, B)$  was selected, then the ontology consisted of the original ChEBI module plus axioms asserting an individual that instantiates  $A$ , but not  $B$ . (Disjointness was used analogously.) Using this approach 1261 inconsistent ontologies were created. Thus, the experiment involved 1429 ontologies. In all cases VAMPIRE evaluation of the consistency of the translated ontology matched the original OWL ontology.

In conclusion, the conducted experiments support that consistency of an ontology is preserved across the translation.

## 6. Related Work

FOWL translates OWL 2 DL axioms into first-order axioms in TPTP format [8]. The Syntax of OWL 2 DL is specified in [21]. FOWL’s translation is based on OWL’s Direct Semantics [2],

<sup>7</sup><http://robot.obolibrary.org/>

which is itself based on the description logic SROIQ [3]. The comparison of OWL and FOL reasoning was investigated in [22, 23]. These papers include a discussion of the translation from OWL DL, which is based on the description logic SHOIND<sub>n</sub><sup>-</sup>, to FOL. The translation follows the model theoretic semantics of SHOIND<sub>n</sub><sup>-</sup>, datatypes are translated as predicates and data values as constants. FOWL applies the same design decisions to the translation of OWL 2 DL. As far we can tell, this previous implementation of the translation from OWL DL to FOL has not been made available publicly.

The Heterogeneous Tool Set, Hets, supports the translation of OWL 2 DL ontologies into FOL theories in various a number of formats including TPTP [24].<sup>8</sup> One significant difference between FOWL and Hets is that Hets is a stand-alone tool written in Haskell, which is feature-rich with a wide range of functionality. FOWL, in contrast, is specifically developed for the task of translating OWL to FOL, and is therefore more lightweight and easy to use and install. FOWL is deployed as a Python library and, thus, is easier to integrate in Python tool chains.

## 7. Conclusion and Future Work

Enabling the translation of OWL ontologies into first order logic gives ontology designers the ability to combine the strengths of both languages. The OWL language is relatively easy to use and is well supported by tools, thus it has been widely adopted for ontology engineering projects. Nevertheless, it is common that nuances and axiomatic complexities of domain knowledge can remain outside the reduced expressiveness of OWL as compared to the full FOL. The presented tool can translate OWL ontologies into FOL in order to enable modular addition of and reasoning with FOL-native axioms.

The tool allows export of the translation result in TPTP syntax. Additional conjectures can then be added, using TPTP syntax, to be reasoned over in combination with the translation, and the ontology can be checked for consistency both in OWL and in FOL. FOWL has been tested on different ontologies and has been shown to preserve the consistency properties of ontologies during translation.

Further work will include improving the performance of the tool as well of adding features of convenience. For example, currently the predicate symbols of the FOL translations are based on the IRIs in the OWL file. This works well for ontologies that use meaningful IRIs, but is inconvenient for ontologies that use alphanumeric IRIs as class and property names. Hence, we are planning to provide an option that uses a given annotation property (like *rdfs:label*) for the generation of predicates. In addition, we will define a vocabulary for annotating OWL ontologies with FOL axioms. FOWL will recognise these annotations and enrich the OWL translation with these additional FOL axioms.

Although, in general, FOL model finders do not scale well, the performance may be significantly increased by techniques like optimal definition elimination [25]. Since the ontologies generated by FOWL only utilise a small fragment of the expressivity of FOL, it is an open question how FOL model finders perform on ontologies that FOWL creates. We furthermore plan to check the consistency of ontologies using BFO as an upper level with the BFO FOL version.

---

<sup>8</sup><http://www.hets.eu/>

## References

- [1] OWL 2 Web Ontology Language Document Overview (Second Edition), 2012. URL: <https://www.w3.org/TR/owl2-overview/>.
- [2] B. Motik, P. F. Patel-Schneider, B. C. Grau, I. Horrocks, B. Parsia, U. Sattler, Owl 2 Web Ontology Language Direct Semantics (Second Edition), 2012. URL: <https://www.w3.org/TR/owl2-direct-semantics/>.
- [3] I. Horrocks, O. Kutz, U. Sattler, The Even More Irresistible SROIQ., KR 6 (2006) 57–67.
- [4] M. Horridge, P. F. Patel-Schneider, OWL 2 Web Ontology Language Manchester Syntax (Second Edition), 2012. URL: <https://www.w3.org/TR/owl2-manchester-syntax/>.
- [5] S. Borgo, C. Masolo, Foundational Choices in DOLCE, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 361–381. doi:10.1007/978-3-540-92673-3\_16.
- [6] R. Arp, B. Smith, A. D. Spear, Building Ontologies with Basic Formal Ontology, MIT Press, 2015.
- [7] D. Osumi-Sutherland, S. Reeve, C. J. Mungall, F. Neuhaus, A. Ruttenberg, G. S. X. E. Jefferis, J. D. Armstrong, A strategy for building neuroanatomy ontologies, Bioinformatics 28 (2012) 1262–1269. doi:10.1093/bioinformatics/bts113.
- [8] G. Sutcliffe, The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0, Journal of Automated Reasoning 59 (2017) 483–502.
- [9] L. Kovács, A. Voronkov, First-Order Theorem Proving and VAMPIRE, in: International Conference on Computer Aided Verification, Springer, 2013, pp. 1–35.
- [10] S. Schulz, E – a Brainiac Theorem Prover, AI Communications 15 (2002) 111–126.
- [11] M. Schneider, J. Carroll, I. Herman, P. F. Patel-Schneider, OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition), 2012. URL: <https://www.w3.org/TR/owl2-rdf-based-semantics/>.
- [12] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, HermiT: an OWL 2 Reasoner, Journal of Automated Reasoning 53 (2014) 245–269.
- [13] M. Booshehri, L. Emele, S. Flügel, H. Förster, J. Frey, U. Frey, M. Glauer, J. Hastings, C. Hofmann, C. Hoyer-Klick, et al., Introducing the Open Energy Ontology: Enhancing Data Interpretation and Interfacing in Energy Systems Analysis, Energy and AI (2021) 100074.
- [14] A. Bandrowski, R. Brinkman, M. Brochhausen, M. H. Brush, B. Bug, M. C. Chibucos, K. Clancy, M. Courtot, D. Derom, M. Dumontier, et al., The Ontology for Biomedical Investigations, PLOS ONE 11 (2016) e0154556.
- [15] M. Bennett, The Financial Industry Business Ontology: Best Practice for Big Data, Journal of Banking Regulation 14 (2013) 255–268.
- [16] M. Costa, S. Reeve, G. Grumbling, D. Osumi-Sutherland, The Drosophila Anatomy Ontology, Journal of Biomedical Semantics 4 (2013) 1–11.
- [17] C. J. Mungall, C. Torniai, G. V. Gkoutos, S. E. Lewis, M. A. Haendel, Uberon, an Integrative Multi-Species Anatomy Ontology, Genome Biology 13 (2012) 1–20.
- [18] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al., Gene Ontology: Tool for the Unification of Biology, Nature Genetics 25 (2000) 25–29.
- [19] S. Carbon, E. Douglass, B. M. Good, D. R. Unni, N. L. Harris, C. J. Mungall, S. Basu, R. L.

- Chisholm, R. J. Dodson, E. Hartline, et al., The Gene Ontology Resource: Enriching a GOLD Mine, *Nucleic Acids Research* 49 (2021) D325–D334.
- [20] K. Degtyarenko, P. De Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcántara, M. Darsow, M. Guedj, M. Ashburner, ChEBI: a Database and Ontology for Chemical Entities of Biological Interest, *Nucleic Acids Research* 36 (2007) D344–D350.
- [21] B. e. a. Motik, *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*, 2012. URL: <https://www.w3.org/TR/owl2-syntax/>.
- [22] D. Tsarkov, I. Horrocks, DL Reasoner vs. First-Order Prover., in: D. Calvanese, G. De Giacomo, E. Franconi (Eds.), 2003 International Workshop on Description Logics (DL2003), volume 81, CEUR, 2003. URL: <http://ceur-ws.org/Vol-81/>.
- [23] D. Tsarkov, A. Riazanov, S. Bechhofer, I. Horrocks, Using VAMPIRE to Reason with OWL, in: S. A. McIlraith, D. Plexousakis, F. van Harmelen (Eds.), *The Semantic Web – ISWC 2004*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 471–485.
- [24] T. Mossakowski, C. Maeder, K. Lüttich, The Heterogeneous Tool Set, Hets, in: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2007, pp. 519–522.
- [25] S. Stephen, T. Hahmann, Model-finding for externally verifying FOL ontologies: A study of spatial ontologies, in: B. Brodaric, F. Neuhaus (Eds.), *Formal Ontology in Information Systems - Proceedings of the 11th International Conference, FOIS 2020, Cancelled / Bozen-Bolzano, Italy, September 14-17, 2020*, volume 330 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 233–248. URL: <https://doi.org/10.3233/FAIA200675>. doi:10.3233/FAIA200675.