# When one Logic is Not Enough: Integrating First-order Annotations in OWL Ontologies

Simon Flügel [a], Martin Glauer [a], Fabian Neuhaus [a,*] and Janna Hastings [b,c]
[a] *Institute for Cooperating Systems, Otto von Guericke University Magdeburg, Germany*
[b] *Faculty of Medicine, Institute for Implementation Science in Health Care, University of Zurich, Switzerland*
[c] *School of Medicine, University of St Gallen, Switzerland*

**Abstract.** In ontology development, there is a gap between domain ontologies which mostly use the web ontology language, OWL, and foundational ontologies written in first-order logic, FOL. To bridge this gap, we present Gavel, a tool that supports the development of heterogeneous 'FOWL' ontologies that extend OWL with FOL annotations, and is able to reason over the combined set of axioms. Since FOL annotations are stored in OWL annotations, FOWL ontologies remain compatible with the existing OWL infrastructure. We show that for the OWL domain ontology OBI, the stronger integration with its FOL top-level ontology BFO via our approach enables us to detect several inconsistencies. Furthermore, existing OWL ontologies can benefit from FOL annotations. We illustrate this with FOWL ontologies containing mereotopological axioms that enable new meaningful inferences. Finally, we show that even for large domain ontologies such as ChEBI, automatic reasoning with FOL annotations can be used to detect previously unnoticed errors in the classification.

Keywords: ontology, heterogeneous ontology, first-order

## 1. Introduction

The landscape of applied ontology contains a rift. The vast majority of ontologies that are used in computational systems are written in OWL, more specifically OWL 2 DL or OWL language profiles, which are essentially sub-languages of OWL 2 DL.[1] However, the authors who work on the foundational topics of applied ontology, such as upper level entities and their axiomatisations, typically do not use OWL. E.g., in 2020 and 2021 the *Formal Ontology in Information Systems* conferences accepted altogether 19 papers on foundational topics. Of those, none used OWL as representational language.

The reason for this language rift is easy to explain. The OWL Manchester Syntax is relatively easy to learn for domain experts, and there are an increasing number of widely available tools supporting development of ontologies in the OWL language. Furthermore, OWL is based on the description logic SROIQ and, thus, satisfiability of OWL ontologies and logical inference with OWL ontologies are decidable decision problems. Both factors are major advantages for anybody who develops a domain ontology in cooperation with domain experts and intends to use automatic reasoning (e.g., using consistency checks for quality control). However, the advantages come with a price: in order to improve readability and achieve decidability one needs to sacrifice expressivity. Consequently, many ontological differences that are studied by foundational ontologists are not expressible in OWL. E.g., part-

*Corresponding author. E-mail: fneuhaus AT ovgu.de.

[1] In the following we will for the sake of brevity refer to OWL 2 DL and OWL 2 language profiles as 'OWL'. We are aware of so-called 'OWL 2 Full' (i.e., OWL 2 without DL constraints and a RDF-based semantics), but since it is rarely used, we will not consider it in this paper.

whole relationships are a staple of most ontologies and have been extensively studied by foundational ontologists. However, in OWL 2 DL it is even impossible to axiomatise proper parthood as strict order, let alone represent interesting distinctions between different mereologies [1]. Hence, foundational ontologists typically use first-order logic (FOL) or even more expressive logics as representational languages. Consequently, the vast majority of papers on foundational ontology contain axioms and definitions that cannot be directly reused by the ontologies that are used in practice in information systems. The language rift thus prevents theoretical results and insights from achieving their full impact in benefiting downstream practical applications.

A particularly interesting case are upper level ontologies, which aim to provide a reusable framework for the development of domain ontologies. There are a number of different upper level ontologies [2], which reflect a wide range of philosophical points of view. Most widely used is the Basic Formal Ontology (BFO), which is used by more than 250 ontology projects. BFO is grounded in a rich Aristotelian tradition, is described in numerous publications, a user guide, and a textbook, and most recently it has even been standardised by the International Organization for Standardization as ISO/IEC 21838-2:2021. However, the version of BFO that is actually used in most information systems is BFO 2.0 OWL, which contains exactly 52 axioms, more precisely 18 disjointness axioms and 34 subsumption axioms between atomic classes. Interestingly, BFO 2.0 OWL contains annotations that include additional axioms in the Common Logic Interchange Format (CLIF), a variant of FOL[2], which provide a much richer logical axiomatisation than the OWL axioms. However, since OWL reasoners ignore annotations, these axioms are computationally inert; they only serve as documentation for the human readers. In summary, while there is a rich literature on BFO, almost none of it is materialised in the format that is actually used by domain ontologies. From a logical point of view BFO 2.0 OWL is just a simple taxonomy with a few additional disjointness axioms. Consequently, the developers of domain ontology cannot use OWL reasoners or other automatic reasoners to check whether their ontology actually conforms to BFO.

The goal of this paper is to present *Gavel* [3] and its OWL-specific extension *Gavel-OWL* [4], tools that were developed to bridge the language rift by enabling ontology developers to develop heterogeneous ontologies that contain both FOL and OWL axioms, and supports reasoning with the integrated model. One major benefit of *Gavel* and *Gavel-OWL* is that it does not require developers of applied ontologies to make any changes to their established workflow for OWL ontologies: developers can continue to use the same tools (e.g., Protégé) to develop their ontologies, and use standard OWL reasoners to reason with the OWL part of the ontology. *Gavel-OWL* merely offers the additional option to enhance an OWL ontology with FOL axioms. These additional axioms may be based on a foundational ontology, but they may also be domain specific axioms that are expressible in FOL but not in OWL.

In section 2 that follows, we discuss the requirements for *Gavel-OWL* and our approach for addressing them. The capabilities of *Gavel-OWL* and its implementation are presented in section 3. Afterwards, we discuss in section 4 the merits of the approach by first showing how *Gavel-OWL* overcomes some limitations of OWL that have been discussed in the literature, and then we consider two case studies. In the first, we will show that BFO CLIF axioms may be used to find errors in an BFO-based OWL domain ontology. In the second, we extend a large domain ontology with FOL axioms and illustrate how automatic theorem proving with the heterogeneous ontology yields new subsumptions that an OWL reasoner on its own is not able to detect. In the remainder of the paper we discuss related work, the significance and potential impact of the approach.

## 2. Requirements and Approach

Our goal is to enable ontology developers to benefit from the advantages of OWL, including its user friendly syntax, decidability, and mature tool chain, without having to forego the ability to use the greater expressivity of FOL when it is required, or when a preexisting FOL ontology is available.

Thus, a *FOWL ontology* consists of two components, an OWL component and a FOL component, which extends the OWL axioms with additional axioms in FOL. To achieve this goal our solution ideally should meet the following requirements:

[2]Technically, Common Logic is more expressive than FOL [3], but for the purpose of this paper the differences may be ignored.
[3]https://github.com/gavel-tool/python-gavel
[4]https://github.com/gavel-tool/python-gavel-owl

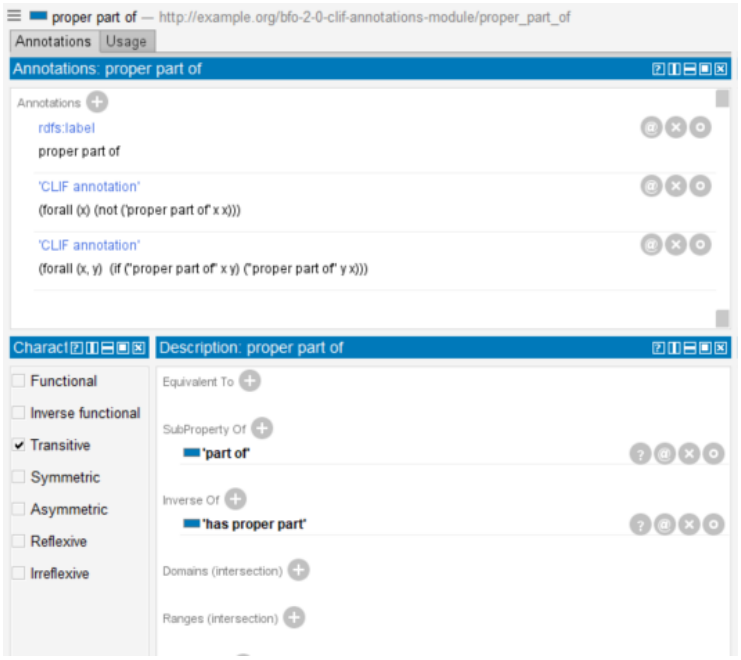


Figure 1. View of an annotated OWL entity in Protégé.

1. In order to remain accessible by the existing OWL infrastructure, FOWL ontologies should be syntactically valid OWL ontologies.
2. The OWL component of a FOWL ontology should be usable by existing OWL tools and follow the OWL 2 DL semantics as specified in [4].
3. For convenience of editing and debugging, it should be possible to associate FOL axioms closely with related OWL entities and axioms. Ideally, this should be able to be done with tools like Protégé, which are widely used for ontology editing.
4. In addition, it should be possible to integrate existing FOL ontologies (e.g., a foundational ontology) with a FOWL ontology.
5. Automatic reasoning with an FOWL ontology should utilise both its OWL and its FOL axioms.

Our approach for meeting these requirements is to create a combined OWL and FOL ontology through annotations in OWL ontologies. Thus, in some sense we follow the precedent set by BFO 2.0 OWL, which is also an OWL file that contains FOL annotations. However, while in BFO 2.0 OWL the FOL annotations are logically inert, *Gavel-OWL* enables automatic reasoning with the logical theory that is the result of combining the OWL axioms with the FOL axioms.

More specifically, a *FOWL ontology* is an OWL file, where one or multiple annotation properties are selected, and all values of *AnnotationAssertion* axioms with these annotation properties are interpreted as FOL axioms. Hence, the heterogenous ontology is represented in a syntactically valid OWL file and the OWL part of the ontology may be processed by standard OWL tools, because the FOL annotations do not influence the ontology's OWL semantics.

As they are stored in annotation values, the FOL axioms also remain closely connected to the OWL axioms: typically, all axioms for a given subject entity are grouped together. For instance, Figure 1 shows how Protégé displays annotations and logical OWL axioms for the entity *proper part of*. The FOL axioms can be seen in the upper part and the OWL axioms in the lower part. Thus, a user is able to view related FOL and OWL axioms on the same screen.

*Gavel-OWL* supports two different syntaxes for FOL annotations: CLIF and TPTP. CLIF is a human readable syntax for ISO Common Logic [3].[5] The TPTP syntax is widely used in the automatic theorem proving community, in particular the TPTP problem library [5, 6]. To increase the readability of TPTP, we do not use complete TPTP axioms, but only their formula part. E.g., we write

```
![X] ~ 'proper part of' (X,X)
```

instead of the full TPTP expression

```
fof(axiom_name, axiom, (![X] ~ 'proper part of' (X,X))).
```

Reasoning with a FOWL ontology requires the integration of its OWL component with its FOL component. This is done by translating the OWL axioms into FOL (in TPTP syntax) and integrating them with the FOL axioms from the annotations[6] as well as, potentially, additional axioms from a background theory. The result is a FOL ontology in TPTP syntax, which can be used for automatic reasoning and consistency checking. To enable these automated consistency checks, *Gavel* is able to interact with the theorem prover Vampire [7]. Furthermore, *Gavel* features a dynamic extension scheme that allows for the integration of other logics through new plugins. We present here *Gavel-OWL*, which allows the integration of first-order logic and OWL. Other ontologies may also use logics for annotation, such as typed logics or higher order logics. These can then be connected to *Gavel* by separate plugins similar to *Gavel-OWL* and to use the same proof pipelines described in this submission.

One obstacle for the integration of OWL axioms and FOL annotations is that it is necessary to map their signatures. E.g., in the example in Figure 1 an OWL entity, more specifically an object property, is annotated with a CLIF axiom. The OWL and the FOL axioms together specify proper parthood as strict partial order. (This could not be axiomatised in OWL 2 DL, since the axiom would violate its global restrictions.) However, strictly speaking, the OWL axioms are about the entity uniquely identified by a long IRI (namely, http://purl.obolibrary.org/obo/BFO_0000175) , which is then annotated with the literal 'proper part of'^^xsd:string as label, while the CLIF axioms are using neither the OWL IRI nor the exact label of the OWL entity, but the CLIF name "proper_part_of". This is indicative of a typical problem. IRIs are often too long to be conveniently typed by human. Thus, one cannot expect the FOL annotations to include the complete IRIs that are used to identify an OWL entity. Further, following recommended best practices [8], many OWL ontologies contain IRIs that consists of arbitrary alpha-numeric identifiers, thus, they are not human readable. In addition, CLIF and TPTP have both restrictions on their syntax. Hence, typically, there are no exact matches between the signatures of the OWL ontology and the signature of the FOL axioms. *Gavel-OWL* attempts to automatically resolve these issues by considering suffixes of IRIs and the labels of OWL entities, when trying to match OWL entities to FOL symbols.

All in all, our methodology provides a practical, user-friendly approach to combining the advantages of both OWL and FOL: By keeping the main part of the ontology in OWL and only adding FOL axioms as annotations, the ontology can be used as a standard OWL ontology. This means, that developers can make use of the existing tools, reasoners and their familiarity with the language. The FOL annotations can then be added to extend the ontology with axioms that are not expressible in OWL, increasing the possibilities developers have for modelling their domain as accurately as possible and allowing the integration of FOL upper level ontologies. Reasoning support for the complete ontology is ensured by *Gavel-OWL* via translating into a unified FOL theory.

## 3. Capabilities and Implementation

To achieve its objective of supporting the development of heterogeneous ontologies that consist of an OWL file with FOL annotations (either in CLIF or TPTP syntax), *Gavel-OWL* has the following capabilities:

[5] Strictly speaking, CLIF's syntax is more flexible than 'normal' FOL because it does not distinguish between individual constants, predicate and function symbols. In addition, CLIF includes sequence variables which increase the expressiveness of CLIF beyond FOL. However, for the purpose of this paper we treat CLIF just as an alternative syntax for 'normal' FOL.

[6] CLIF axioms are also translated into TPTP syntax.

1. OWL ontologies can be translated into a file in TPTP syntax.
2. OWL ontologies that are annotated with FOL axioms (in either CLIF or TPTP) syntax can be translated into a file in TPTP syntax, which integrates the translation of the OWL axioms with the FOL axioms in the annotations. The signatures are harmonised, optionally using IRIs or human friendly identifiers.
3. OWL ontologies that are annotated with FOL axioms can be translated into the Distributed Ontology, Modelling and Specification language (DOL), where the OWL component and the FOL component of the heterogeneous ontology are represented as two different sub-ontologies.
4. OWL ontologies can be checked for consistency via the OWL reasoner HermiT [9].
5. An automatic theorem prover for FOL (currently, Vampire) may be used to attempt to prove FOL conjectures (in TPTP syntax) from an OWL ontology (possibly with FOL annotations).
6. Alternatively to (5), the conjectures may also be provided by a second OWL ontology. In this case *Gavel-OWL* attempts to prove that the second ontology is logically entailed by the first. This is implemented by translating the axioms of the second ontology into FOL and using them as conjectures of TPTP problems.

The *Gavel-OWL* implementation builds on two pre-existing components. First, the **OWL API** [10] [7] is an open source Java library for working with OWL ontologies. It has originally been developed by the University of Manchester and is currently maintained by Matthew Horridge and Ignazio Palmisano. We use it for parsing OWL ontologies and extracting FOL annotations. Since the OWL API is a Java library, *Gavel-OWL* includes a *Java Server* that handles the interaction with the OWL API and contains large parts of the OWL-related logic. It is connected to the application's Python part via Py4J [8]. Second, **Macleod** [9] is a Python tool for working with ontologies written in the CLIF syntax, created by Torsten Hahmann. We use it to translate CLIF annotations into TPTP syntax.

Figure 2 illustrates the implementation of the main functionality of *Gavel-OWL*, namely the translation of an OWL ontology (possibly with FOL annotations) to FOL. In the case of an OWL ontology without any FOL annotations the following things happen: First, it parses the OWL ontology by using the OWL API and subsequently translates it into FOL. Optionally, IRIs are replaced with more readable names. The translation from OWL to FOL, previously described in [11], is based on the OWL Direct Semantics [4] and uses a mapping $[\quad]_q^p$ between OWL and FOL expressions.[10] This mapping is defined for each basic type of OWL axiom or expression and is then applied recursively on more complex axioms. In doing so, the parameters $p$ and $q$ are used to keep track of substitutions that have to be made and are left out if not needed. E.g., $SubclassOf(Fish, Animal)$ would be translated as follows.

$$[SubclassOf(Fish, Animal)] \quad \Leftrightarrow \quad \forall x([Fish]^x \rightarrow [Animal]^x) \quad \Leftrightarrow \quad \forall x(Fish(x) \rightarrow Animal(x))$$

A FOL translation of a complete OWL ontology consists of the translations for each axiom combined with additional background axioms. These capture general assumptions of the Direct Semantics (e.g., the distinction between object and data domains) and the meaning of the OWL reserved vocabulary, such as `owl:Nothing`. These steps are implemented in the Java Server, because they build upon the OWL API functionality. Therefore, the resulting FOL axioms need to be transferred back to the Python OWL Parser and transformed into Gavel's internal representation. This representation is then used by Gavel to compile it into a TPTP document, which gets returned to the user.

OWL files with FOL annotations are split into three parts. The OWL part of the ontology is translated into the internal FOL representation of Gavel as described above. TPTP annotations are extracted, turned into valid TPTP expressions and then parsed by Gavel. CLIF is parsed with the help of Macleod, which has been modified to return Gavel's internal representation as well. In order to align the FOL signature with the OWL signature, the Java Server first fetches the OWL signature and *rdfs:label*-values and then maps them to the FOL names using the Levenshtein distance. If no close match is available, the name will be used as-is, making the assumption that there is no OWL entity to be found and that this name only appears in FOL axioms. Based on this mapping FOL names are replaced with IRIs and in a second step, if required, with readable names. Finally, the FOL axioms from annotations are combined with the FOL translation of the OWL part, which also uses either IRIs or readable names, and compiled to TPTP, completing the translation process.

[7] https://github.com/owlcs/owlapi
[8] https://www.py4j.org/
[9] https://github.com/thahmann/macleod
[10] A detailed specification of the mapping is available at https://github.com/gavel-tool/python-gavel-owl/wiki/OWL-to-FOL-mapping.

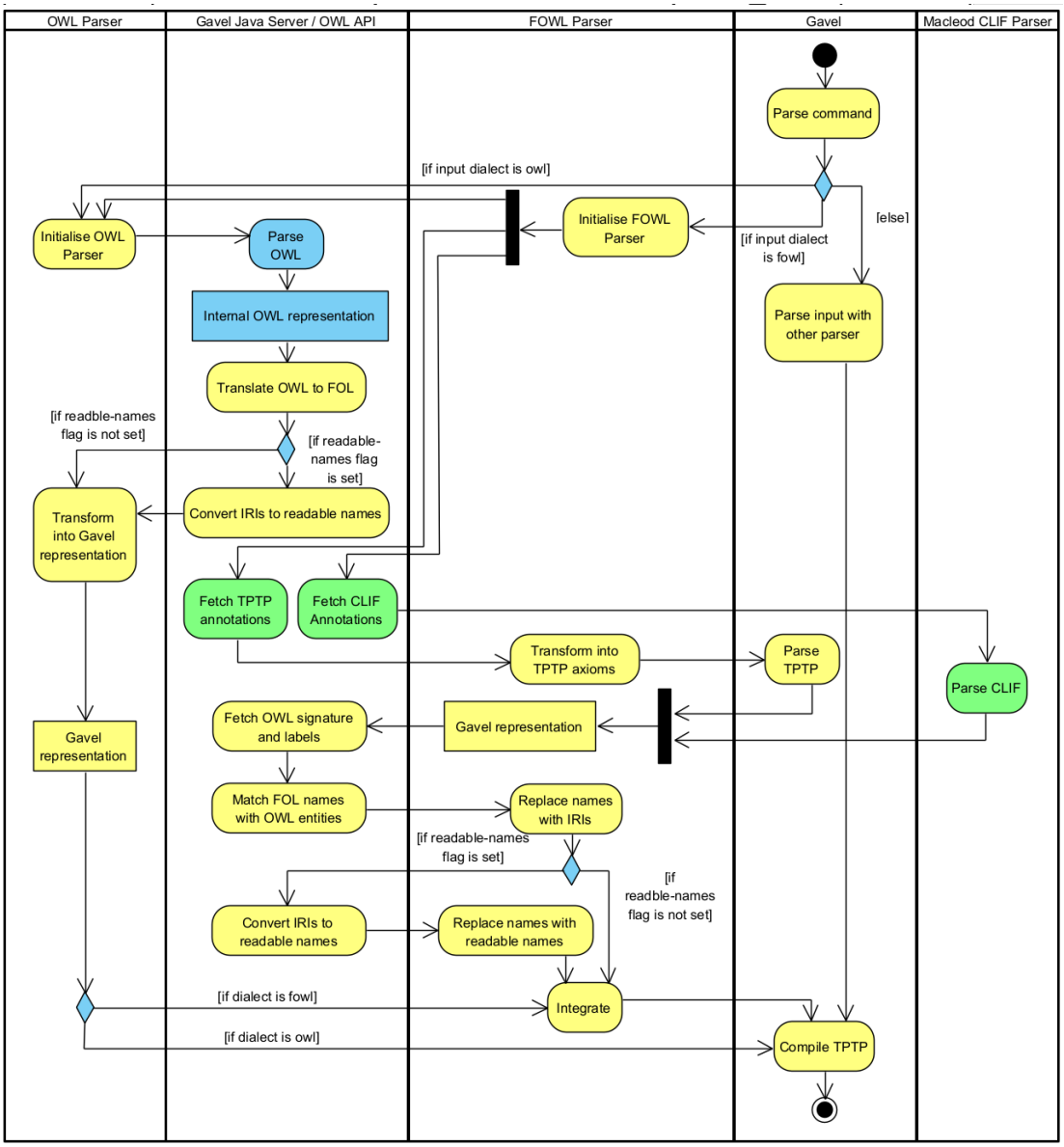


Figure 2. A UML activity diagram illustrating the process of translating an annotated OWL ontology or standard OWL ontology into TPTP. Actions that were newly implemented for the translation tasks are shown in yellow, actions that were reused from existing tools are marked in blue. For green actions, the existing functionality has been extended.

## 4. Evaluation and Use Cases

We aim to evaluate *Gavel-OWL* in use for ontology development. For this purpose, we first evaluate whether *Gavel-OWL* helps to overcome OWL restrictions *that are ontologically meaningful*. Since *Gavel-OWL* enables reasoning with arbitrary FOL axioms, it also enables inferences with FOWL ontologies that an OWL reasoner would not be able to support. However, it is not obvious whether this additional expressivity is also useful. Thus, we evaluate the capability of *Gavel-OWL* based on examples from the literature in which the authors propose ontologically meaningful modelling patterns that cannot be expressed in OWL.

As mentioned in the introduction, one motivation for developing *Gavel-OWL* is to support a better integration of upper level ontologies with domain ontologies. Thus, as a second way to show the benefits of *Gavel-OWL*, we developed a new version of BFO that integrates OWL and FOL, and used it to check consistency of the Ontology for Biomedical Investigations (OBI) [12], a widely used ontology in the life sciences.

In another case study we show how domain specific FOL axioms may be used to increase the quality of a domain ontology. For this purpose we extended the Chemical Entities of Biological Interest (ChEBI) ontology [13] with roughly 120 000 FOL axioms. Gavel allowed us to detect errors in ChEBI, which OWL reasoning alone was not able to detect. This example illustrates that even large domain ontologies may benefit from FOL axioms.

In a previous publication [11] we discussed the performance of the translation from OWL to TPTP (e.g., time per translated axiom). We found that even large ontologies may be translated in a reasonable amount of time (e.g., the Gene Ontology is translated in less than 30 minutes on a normal laptop). However, since there is no reason why one would translate the same large ontology regularly, performance may not be the most important metric.

### *4.1. Evaluation via Ontologically Meaningful Inferences*

For the purpose of this evaluation we consider modelling patterns that have been identified in the literature as ontologically meaningful, but which cannot be expressed in OWL. The first examples are mentioned in a paper by Schneider, Rudolph and Sutcliffe on a version of OWL 2 DL without global restrictions [14]. In order to show the advantages of their unrestricted OWL version, they provide 12 examples of modelling patterns that are useful but are not syntactically valid OWL 2 DL, since they violate its global restrictions. For each example they also provide a conjecture ontology, which is logically entailed according to the OWL Direct Semantics, but cannot be inferred by OWL reasoners because of the global constraint violation. For our purposes we replaced the OWL axioms that violate the DL restrictions with logically equivalent FOL annotations. Thus, the result are twelve pairs of ontologies, each pair consisting of a FOWL premise ontology representing a useful modelling pattern and a OWL conjecture ontology that is entailed by the premise ontology, but cannot be inferred by OWL 2 DL reasoners. In each of the 12 examples, *Gavel-OWL* is able to prove that the conjecture ontology is logically entailed by the FOWL premise ontology.

A second set of test cases has been derived from a paper by Keet et al. on mereotopological relations [1], in which the KGEMT mereotopological theory [15] is used to strengthen the axiomatisation of ontologies and thereby improve their accuracy. One of the challenges Keet et al. face is the limited expressiveness of OWL: Of the 27 definitions and axioms they identified in KGEMT only 9 can be fully represented in OWL 2 DL, but 3 of these can be only be represented individually, but not collectively [11].

For the purposes of our evaluation, we can use the set of 27 axioms in [1] as a basis for generating test cases. Out of the 27 axioms, only 3 are not representable in FOL, but require Higher Order Logic (HOL). Thus, nearly 90% of the axioms are representable in either OWL or FOL (see Figure 3). The 15 axioms which are representable in FOL, but not in OWL, were used for creating test cases. An additional test case is based on the axioms which cannot be represented collectively in OWL, leading to a total of 16 test cases.

Each test case consists of a FOWL premise ontology which consists mostly of OWL axioms but includes one of the KGEMT axioms as a FOL annotation, and an OWL conjecture ontology, which cannot be inferred by OWL reasoning from the premise ontology. For example, the premise ontology in Listing 1 contains OWL axioms that express that every part of OVGU is part of Magdeburg. Further, it includes a KGEMT axiom as a TPTP annotation, which states: for all x and y, if y (e.g., OVGU) is not part of x (e.g., Magdeburg), there has to be an instance z that is part of y, but does not overlap x. The conjecture ontology of this example states that OVGU is part of Magdeburg.

In each of the 16 test cases *Gavel-OWL* is able to prove that the conjecture ontology is entailed from the premise ontology. This result illustrates that though some axioms had to be left out because they are not representable in FOL, the axiomatisation of an ontology covering mereotopological concepts can become significantly stronger when using FOL annotations. More generally, the results of both test cases indicate our tool is working correctly and that it is able to integrate the FOL annotations and the ontology's OWL part into a coherent logical theory that can be

[11] These three axioms refer to the irreflexivity, asymmetry and transitivity of *proper part of*, because OWL's global restrictions prohibit the combination of these characteristics for the same object property.

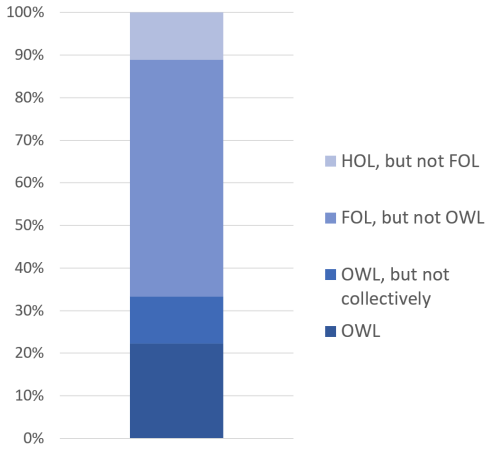


Figure 3. The shares of KGEMT axioms which can be represented in OWL, FOL and HOL respectively.

```
Class: ovgu_part
    EquivalentTo:
        part_of value ovgu
    SubclassOf:
        overlap value magdeburg

ObjectProperty: overlap
ObjectProperty: part_of
    Annotations:
        tptp_annotation "![X, Y]: (~
    part_of(Y, X) => ?[Z]: (part_of(Z,
    Y) & ~overlap(Z, X)))"

Individual: ovgu
Individual: magdeburg
```

Listing 1 The premise ontology of a test case based on a KGEMT axiom.

used for reasoning tasks. Further, they illustrate that the additional expressivity granted by FOL annotations may be used to address some of the limitations of OWL that have been discussed in the literature. All test cases can be found on the project's GitHub page [12].

### *4.2. Extending BFO*

With FOWL, we can not only add individual FOL axioms to OWL ontologies, but also integrate whole FOL and OWL ontologies. This is especially useful where OWL domain ontologies are based on FOL upper level ontologies. To demonstrate this, we used the BFO in combination with the Ontology for Biomedical Investigations (OBI) [12]. OBI uses BFO 2.0 OWL, which due to restrictions in the OWL language does not represent the complete axiomatisation of BFO. The latter is only available in FOL. Thus, it is not able to check if OBI fully conforms to BFO with an OWL reasoner.

BFO 2.0 OWL does however already contain annotations containing axioms from BFO 2.0 FOL in CLIF syntax. We used these annotations to build a new BFO version where the annotations are interpretable by CLIF parsers and can be combined with BFO 2.0 OWL as well as OWL domain ontologies. For that, we had to make several changes: Firstly, the annotations in BFO 2.0 OWL contain additional comments besides the CLIF axioms which we removed. Secondly, the axioms in BFO 2.0 FOL are temporalized, leading to ternary predicates which are not compatible with OWL ontologies. Therefore, we removed this temporalization. This is based on the assumption that OWL ontologies in general make time-independent statements.

Additionally to the FOL axioms, our version of BFO contains OWL object properties from the Relation Ontology (RO) [13] and BFO 2020 OWL [14], another BFO version, corresponding to the binary predicates in our modified BFO 2.0 FOL axioms. This is necessary because BFO 2.0 OWL does not contain any object properties, just the BFO classes. Where possible, we used the object properties from RO, because they are already used by many BFO-based ontologies, including OBI. In addition, we have adapted the names of binary FOL predicates to match the object properties from RO and BFO 2020 OWL where necessary. This ensures that *Gavel-OWL* can match both types of relations, FOL predicates and OWL object properties, correctly. Similarly, for the unary predicates, we replaced the BFO CLIF labels with the BFO OWL labels as they are specified in BFO 2.0 OWL. Lastly, we imported our BFO version[15] into OBI and used *Gavel-OWL* to translate the resulting ontology to FOL. Subsequently, we checked the

[12] https://github.com/gavel-tool/python-gavel-owl
[13] https://github.com/oborel/obo-relations/
[14] https://github.com/BFO-ontology/BFO-2020
[15] The FOWL version of BFO can be accessed at https://github.com/gavel-tool/BFOnow

FOL theory for satisfiability, which correlates with the ontology's internal consistency. Internal consistency means that a model can be constructed for the ontology, compared to external consistency, which signifies that a model can be constructed from datasets that represent an ontology's intended use [16]. We checked the latter one by adding one individual to every OBI-class and repeating the process. Here, the intended use case we considered is one where all classes can be instantiated.

In total, we were able to detect 4 inconsistencies, 2 of them external inconsistencies. For illustration purposes we discuss one of these inconsistencies in more detail. It arises from a conflict mainly between two axioms, one being BFO 2.0 FOL axiom number 134-001,

```
(forall (x) (if ("independent continuant" x)
                (exists (r) (and ("spatial region" r) ("located in" x r)))),
```

stating that for every *independent continuant*, there is a *spatial region* in which it is located. It is in conflict with an OWL axiom belonging to the OBI class *local field potential recording*,

```
(has_specified_input some
    ((tissue and ('located in' only brain))
        and ('has role' some 'evaluant role'))),
```

which ascertains that a *local field potential recording* is related to an instance of *tissue* that only has a *located in*-relationship to instances of *brain*. Since *tissue* is an *independent continuant*, every instance of it has to be located in some *spatial region*, which is an *immaterial entity*. *immaterial entity* is disjoint with *material entity* of which *brain* is a subclass. Thus, no instance of *tissue* can be *located in only brain* and *local field potential recording* cannot be instantiated.

This example illustrates that *Gavel* can be used to find inconsistencies in existing OWL ontologies that have previously been undetected because of the lack of access to automatic reasoning with FOL axioms.

While using the BFO FOL axioms, we also noticed an error in axiom [062-002], where the arguments of the bearer-of relation have been swapped. This indicates that using automatic reasoning can help detect mistakes both in FOL and OWL ontologies which would have remained unnoticed if the languages had not been connected.

Our BFO version can be applied not only to OBI, but to other BFO-based ontologies as well. It and a version containing only the modified BFO 2.0 FOL axioms and the object properties can be found on GitHub [16].

### *4.3. Application to the Chemistry Ontology ChEBI*

ChEBI [13] is a freely accessible, open OWL ontology for a biologically relevant chemical entities. This mainly includes substances that occur naturally in biological processes or synthetic substances that interact with these processes. Figure 4 shows an example of a ChEBI class with annotations. In addition to the molecular graph and textual descriptions and definitions, ChEBI contains a representation of the molecular class as SMILES specification. SMILES is a representation language that is specially developed for molecular entities, and it allows the direct serialisation of a molecular structure. ChEBI is the largest publicly available ontology for chemical entities and covers, as of August 2022, a total of more than 180,000 classes. The maintenance and further development of such an ontology requires a large, heterogeneous team of ontology and domain experts. To facilitate this process, ChEBI allows third-party annotation and uses a special rule-based automatic classifier, ClassyFire [17]. However, ChEBI is generally quite weakly axiomatised and, to our knowledge, no symbolic methods beyond the standard OWL-DL reasoning are currently used to validate ChEBI.

For our case study we extended ChEBI with more than 129,193 FOL annotations[17]. and used these to validate ChEBI's OWL axioms. Most of these FOL axioms were automatically generated based on SMILES annotations. For the purpose of this case study, we consider two different categories of classes of chemical entities in ChEBI.

The first category contains classes that are associated with SMILES strings that contain unspecified parts. These unspecified parts are represented in the SMILES strings as *'*'* and in the molecular graph as *R*, representing the

[16] https://github.com/gavel-tool/BFOnow

[17] The FOWL extension of ChEBI that contains the these additional annotations may be accessed at https://doi.org/10.5281/zenodo.7038560

attachment at a particular point of a group or sub-structural component. For example, the class *Nitrile* in Figure 4 is incompletely specified, because the "*" might be replaced by a lot of different structures.

In contrast, there are the classes of chemical entities whose structure is completely specified by their SMILES string. These classes occur primarily at the lowest levels of the ontological hierarchy and usually have no subclasses, and if they do, they represent stereoisomers, i.e. the subclasses only differ in spacial orientation and not in structure. For example, *L-2-aminopentanoic acid* in Figure 5 is fully specified, i.e. all molecules that instantiate this class have the same molecular structure.

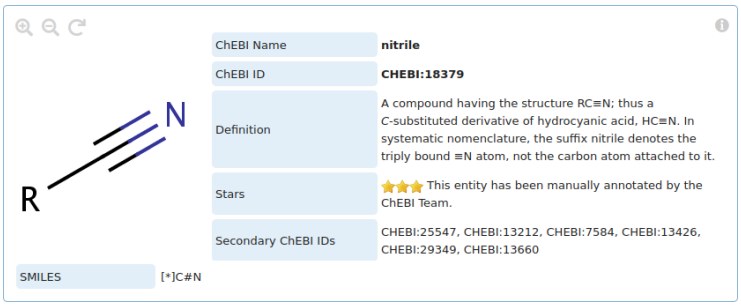


Figure 4. The class *nitrile* as represented on the ChEBI website.

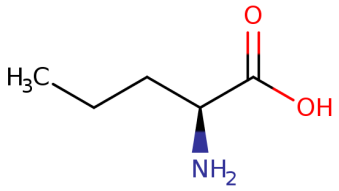


Figure 5. The class *L-2-aminopentanoic acid*, which had been wrongly axiomatised as a subclass of nitrile.

The difference between these two categories of classes is important, since they lead to different kinds of FOL axioms.

Classes that are associated with an incomplete SMILES specification are typically defined via the presence of some substructure. For example, Listing 2 shows an axiomatisation for the molecular class *nitrile*, automatically generated based on its SMILES annotation. (Note that the carbon atom that is bound to the nitrogen is explicitly represented.)

```
fof(chebi_18379_0,
  axiom,
  ![M]:(chebi18379(M) <=> (
    ?[N2, N1]:(  c(N1) & part_of(N1, M) & has_no_charge(N1)
               & n(N2) & part_of(N2, M) & has_no_charge(N2)
               & has_triple_bond_to(N1, N2) & N2 != N1
               & connected(M))))).
```

Listing 2: First order axiomatisation for the molecular class nitrile

In addition, the SMILES annotations of this category of classes often only provide an incomplete representation of the textual definitions available in ChEBI. For example, consider medium- and long-chain fatty acids (CHEBI:59554, CHEBI:15904). Their smiles annotations are identical, but their definitions distinguish them by their respective chains lengths. Medium-chain fatty acids have chains that consist of at least 6 carbon atoms, but not more than 12. Long chains consist of at least 13, but less than 22 carbon atoms. Neither smiles nor OWL DL can express such structures. Another example is classes that are not just defined by the presence but also by the absence

of some substructures, since SMILES has no equivalent to a negation. In some cases where we noticed that there were mismatches or omissions in the generated FOL class definitions, we adjusted the FOL axioms manually.

The formalisation of the second category of classes, i.e., the fully specified ones, is different in two ways. Since all instances of such a class have the same molecular structure, we represent them as a 'prototypical instance'. Hence, their formalisations do not involve quantifiers but individuals. From a logical point of view this is not necessary (we could follow the same pattern of definition as with the other type of classes), but it improves the performance of automatic reasoning significantly. In addition, since the SMILES specification is complete, we need to express that the given molecule contains no other parts than the ones explicitly mentioned in the axioms. Hence, the formalisation of these classes contain a formula of the form of the form `![X]:partOf(X, mol) => (X=n1 | X=n2 | ...)`. For example, the FOL axiom for water (CHEBI:15377) is represented in Listing 3.

```
fof(chebi_15377_inst,
    axiom,
        o(n15377_0) & part_of(n15377_0, m15377) & has_no_charge(n15377_0)
      & h(n15377_1) & part_of(n15377_1, m15377) & has_no_charge(n15377_1)
      & h(n15377_2) & part_of(n15377_2, m15377) & has_no_charge(n15377_2)
      & has_single_bond_to(n15377_0, n15377_1)
      & has_single_bond_to(n15377_0, n15377_2)
      & ~has_bond(n15377_1, n15377_2)
      & ![X]:(part_of(X, m15377)
              => (X=n15377_0 | X=n15377_1 | X=n15377_2))
      & connected(m15377)).
```

Listing 3: First order axiomatisation for the 'prototypical instance' of water

For our experiments we used our extension of ChEBI with these FOL annotations together with an additional ontology that added some chemical background knowledge (e.g., that chemical elements are disjoint). Using *Gavel* we proved that the current version of ChEBI was, in fact, inconsistent. We therefore decided to investigate further and split the ontology into smaller fragments.

For a detailed analysis, we selected a subset of ChEBI consisting of 80 incompletely specified classes and 6569 completely specified classes, each of which is represented by some 'prototypical individual'. For each of these 'prototypical individuals' $a$ and each incompletely specified class $B$, we created the conjecture $B(a)$. Because $a$ is structurally indistinguishable from all other members of its class, the conjecture is $B(a)$ equivalent to the assertion that the class of $a$ is a subclass of $B$.

We used *Gavel* and Vampire to attempt to prove these conjectures and compared the results to the transitive closure of subsumption relations in ChEBI. Assuming that all information in ChEBI is correct, a conjecture $B(a)$ should only be provable from the FOL theory, if $a \sqsubseteq B$ is true in ChEBI. Furthermore, Vampire should only be able to find a counterexample, if $a \sqsubseteq B$ is not an axiom in ChEBI.

The experiments were conducted on a CPU clusters with 80 jobs, using 4 CPUs each. Each job focussed on a single molecular class and attempted to prove membership of all 6569 instances individually, limited by a time out of 30 seconds per proof. Each of these jobs terminated within 24 hours, while 31.06% of all proof attempts were stopped by the time-out. Our experiments showed that the proof complexity scales directly with the complexity of the molecule class. Some complex classes, such as *isoflavones* (CHEBI:38757), exceeded the time-out of 30s on all instances. While the results of the FOL reasoning were mostly consistent with the OWL axioms, there were some interesting findings, which can be distinguished in two categories:

**Unexpected proof** $a \sqsubseteq B$ is not an axiom in ChEBI. Yet, $B(a)$ was provable based on the FOL annotations. This occurred in 10132 proof attempts (0.0279% of all proof attempts).

**Unexpected counter-example** $a \sqsubseteq B$ is an axiom in ChEBI. Yet, Vampire refuted the conjecture $B(a)$. This occurred in 256 proof attempts (0.0007% of all proof attempts).

Unexpected proofs are likely to originate from SMILES annotations that do not specify a class adequately. In these cases the FOL definitions that are automatically generated are too wide, which leads to unexpected proofs. Thus, each of these unexpected proofs is an indication that either a SMILES annotation in ChEBI is erroneous or that SMILES is not expressive enough to represent the definition of the class (e.g., because the class is partially defined by the absence of a feature). As discussed above, in some cases we addressed these unexpected proofs by changing the FOL definitions manually.

Unexpected counter-examples are candidates for inconsistencies in the underlying ontology. For example, the molecule in Figure 5 was classified as a nitrile even though it has a single tetrahedral bond between the carbon atom and the sodium atom and not the triple bond required for nitriles. We reported this finding to the ChEBI developers and it has been acknowledged as an error and fixed in the ontology[18].

Thus, this case study illustrates that FOL annotations may be used to represent knowledge in a large domain ontology, and that reasoning with these FOL annotations with *Gavel* can be successfully used to identify errors in such an ontology.

## 5. Related Work

There are several works on extending the expressivity of OWL ontologies that are closely related to our work. In [14] Schneider et al. present 12 useful modelling patterns that are not expressible in OWL 2 DL, because they violate its global restrictions (e.g., axiomatising an object property as strict partial order). For evaluation purposes these modelling patterns are implemented in OWL ontologies and also translated into FOL. Their experiments show that, while all of the OWL reasoners failed on these test cases, the FOL reasoners were able to draw the desired inferences. Hence, the authors illustrated the feasibility to FOL reasoning for OWL ontologies. However, [14] only considers ontologies that consist of OWL constructs, while our work enables the use of arbitrary FOL annotations as part of OWL ontologies. In that sense our work is more general.

Another important strategy to extend the expressivity of OWL is by adding rules. This is supported by the W3C recommendation Rule Interchange Format (RIF)[18], which supports a variety of dialects based on different logics (e.g., Horn Logic or Datalog), but none of them supports full FOL. Significantly older, but influential is SWRL [19], which combines the OWL and OWL Lite sublanguages with features of the RuleML [20]. It uses Horn-like rules in the form of implications between an antecedent called the body and a consequent called the head. However, the extension of OWL by SWRL is not decidable. For this reason Motik et. al. [21] suggest an alternative, which combines OWL DL with DL-safe rules. These are, roughly speaking, function-free Horn rules, where variables are bound only to named individuals. Even though the combination of OWL and SWRL has a higher expressivity than OWL, it does not provide the same expressivity as FOL. For this reason it was proposed to extend SWRL by arbitrary FOL axioms [22]. However, as far as we are aware, this proposal has never been implemented.

The Distributed Ontology, Modeling, and Specification Language (DOL) [23, 24] is an Object Management Group (OMG) specification, which – among many features – supports the representation of heterogeneous ontologies and translation between ontology languages. DOL is implemented by Hets [25, 26] and supports, in particular, OWL, CLIF, and TPTP. Thus, many capabilities of *Gavel* are also provided by Hets. However, a major difference is the fact that DOL/Hets assumes that a heterogeneous ontology is composed of different sub-ontologies which are written in only one language, e.g., an DOL ontology might import an OWL file and a separate CLIF file. This solution is inconvenient for ontology developers, since the heterogeneous ontology might not be edited by established editors like Protégé and axioms concerning one entity might be spread over different files. Further, *Gavel* provides the capability to automatically align the signatures of the various ontologies. Another difference is that *Gavel* is implemented in Python and is more lightweight than Hets.

[18] https://github.com/ebi-chebi/ChEBI/issues/4281

## 6. Importance, Impact and Conclusions

As discussed in section 1, the field of Applied Ontology is divided by a language barrier. Since ontologists who work on foundational issues typically prefer more expressive languages than OWL, the result of their research cannot be directly applied to ontologies that are written in OWL – which includes the vast majority of all domain ontologies. In particular this is illustrated by the fact that top-level ontologies such as BFO and DOLCE are represented primarily in FOL, but the versions that are actually used by domain ontologies are in OWL. Thus, so far it was not possible to check whether these domain ontologies actually conform to their top-level ontology.

*Gavel* is designed to help ontology developers to bridge the divide, since it enables the development of FOWL ontologies, i.e., OWL ontologies with FOL annotations. In section 4 we have illustrated our approach by validating the Ontology for Biomedical Investigations (OBI). OBI is a relatively small (about 5000 entities), but widely used ontology[19], which uses the (OWL version of) BFO as its top-level ontology. By combining OBI with FOL annotations that were derived from FOL annotations published as part of BFO 2.0 OWL, we were able to detect several inconsistencies.This method of evaluation is not limited to OBI, but is applicable to any of the more than 250 domain ontologies that use BFO as their top-level ontology. By creating analogous FOWL versions of other top-level ontologies, the same strategy may be applied to them as well.

Furthermore, we discussed how *Gavel* is able to reason with a FOWL ontology that extends an OWL ontology with complex mereological axioms. Mereology is one of the best studied areas in applied ontology, but OWL is not expressive enough to represent many mereological distinctions. This example illustrates that *Gavel* enables ontology developers to enhance OWL ontologies with the results of research on foundational ontologies.

The benefit of reasoning with FOWL ontologies with Gavel is not limited to the integration of foundational and domain ontologies. As our third use case illustrates, FOL annotations may be fruitfully used to express domain knowledge, which cannot be expressed in OWL – even in the case of very large ontologies like ChEBI. By applying automatic reasoning we were able to detect a number of potential problems in ChEBI, of which we reported two and they have already been fixed by ChEBI developers. A surprising result is the fact that FOL reasoning with ChEBI is a viable option, even though ChEBI is a very large ontology. Obviously, there were time-outs, but a large majority of proof attempts returned a result and, thus, illustrated the usefulness of the approach.

In conclusion, in this paper we have shown the benefits of FOWL ontologies and presented the tool *Gavel*, which is able to support reasoning with these heterogeneous ontologies. Since these ontologies are syntactically valid OWL ontologies, using these annotations does not break existing tool chains, and, thus, the barrier for using this approach is low.

In the future we are planning to extend our approach from FOL to higher-order logic. This would enable us to cover the mereological axioms in [1] that we had to exclude from our evaluation in section 4. Further, there are definitions of chemical classes that require monadic second-order logic [27].

Another area we are planning to explore is the development of a more densely axiomatised BFO FOWL ontology. For the purposes of this paper we have mainly considered axioms that have been derived from the FOL annotations that are already present in BFO 2.0 OWL. However, these axioms are relatively sparse and could be significantly improved. In combination with Gavel this would enable a better validation of domain ontologies that are based on BFO.

Further, we are planning to integrate our workflow for a logical evaluation of chemical subclass relationships with our work on automatic ontology extension [28–30]. In these works we used machine learning techniques to predict for a given unknown chemical molecule its suitable classification in ChEBI. From a logical point of view, we trained models that – when provided with the structure of a chemical molecule – are able to generate OWL subsumption conjectures. We have already used OWL disjointness axioms to check for some errors in these conjectures. In the future we are planning to use FOL annotations of ChEBI to further automatically validate these conjectures with logical reasoning.

[19] Between July 2021 and July 2022 OBI was downloaded 2163 times from Bioportal. https://bioportal.bioontology.org/ontologies/OBI

## References


[1] C.M. Keet, F.C. Fernández-Reyes and A. Morales-González, Representing Mereotopological Relations in OWL Ontologies with OntoPartS, in: *Extended Semantic Web Conference*, Springer, 2012, pp. 240–254.

[2] S. Borgo, A. Galton and O. Kutz, Special Issue: Foundational ontologies in action, *Applied Ontology* **17-1** (2022), 1–16.

[3] I.O. for Standardization, Common Logic (CL) - A Framework for a Family of Logic-based Languages, Standard, International Organization for Standardization, Geneva, CH, 2018.

[4] I. Horrocks, B. Parsia and U. Sattler, OWL 2 Web Ontology Language Direct Semantics (Second Edition), 2012. https://www.w3.org/TR/owl2-direct-semantics/.

[5] G. Sutcliffe, The TPTP Problem Library and Associated Infrastructure: From CNF to TH0, TPTP v6. 4.0, *Journal of Automated Reasoning* (2017), 1–20.

[6] G. Sutcliffe, S. Schulz, K. Claessen and A. Van Gelder, Using the TPTP Language for Writing Derivations and Finite Interpretations, in: *International Joint Conference on Automated Reasoning*, Springer, 2006, pp. 67–81.

[7] L. Kovács and A. Voronkov, First-order Theorem Proving and Vampire, in: *International Conference on Computer Aided Verification*, Springer, 2013, pp. 1–35.

[8] J.A. McMurry, N. Juty, N. Blomberg, T. Burdett, T. Conlin, N. Conte, M. Courtot, J. Deck, M. Dumontier, D.K. Fellows, A. Gonzalez-Beltran, P. Gormanns, J. Grethe, J. Hastings, J.-K. Hériché, H. Hermjakob, J.C. Ison, R.C. Jimenez, S. Jupp, J. Kunze, C. Laibe, N. Le Novère, J. Malone, M.J. Martin, J.R. McEntyre, C. Morris, J. Muilu, W. Müller, P. Rocca-Serra, S.-A. Sansone, M. Sariyar, J.L. Snoep, S. Soiland-Reyes, N.J. Stanford, N. Swainston, N. Washington, A.R. Williams, S.M. Wimalaratne, L.M. Winfree, K. Wolstencroft, C. Goble, C.J. Mungall, M.A. Haendel and H. Parkinson, Identifiers for the 21st century: How to design, provision, and reuse persistent identifiers to maximize utility and impact of life science data., *PLoS Biology* **15**(6) (2017), e2001414. doi:10.1371/journal.pbio.2001414.

[9] B. Glimm, I. Horrocks, B. Motik, G. Stoilos and Z. Wang, HermiT: an OWL 2 Reasoner, *Journal of Automated Reasoning* **53**(3) (2014), 245–269.

[10] M. Horridge and S. Bechhofer, The OWL API: A Java API for OWL Ontologies, *Semantic Web* **2**(1) (2011), 11–21.

[11] S. Flügel, A. Kleinau, F. Neuhaus, M. Glauer and J. Hastings, FOWL–An OWL to FOL Translator, *Proceedings of the Joint Ontology Workshops 2021* (2021).

[12] 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**(4) (2016), e0154556.

[13] J. Hastings, G. Owen, A. Dekker, M. Ennis, N. Kale, V. Muthukrishnan, S. Turner, N. Swainston, P. Mendes and C. Steinbeck, ChEBI in 2016: Improved services and an expanding collection of metabolites, *Nucleic acids research* **44**(D1) (2016), D1214–D1219.

[14] M. Schneider, S. Rudolph and G. Sutcliffe, Modeling in OWL 2 without Restrictions, in: *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 26-27, 2013*, M. Rodriguez-Muro, S. Jupp and K. Srinivas, eds, CEUR Workshop Proceedings, Vol. 1080, CEUR-WS.org, 2013. http://ceur-ws.org/Vol-1080/owled2013_14.pdf.

[15] M. Aiello, I. Pratt-Hartmann, J. Van Benthem et al., *Handbook of Spatial Logics*, Vol. 4, Springer, 2007, pp. 945–-1038.

[16] S. Stephen and T. Hahmann, Model-Finding for Externally Verifying FOL Ontologies: A Study of Spatial Ontologies, *Frontiers in Artificial Intelligence and Applications* **330** (2020), 233–248. doi:10.3233/FAIA200675.

[17] Y. Djoumbou Feunang, R. Eisner, C. Knox, L. Chepelev, J. Hastings, G. Owen, E. Fahy, C. Steinbeck, S. Subramanian, E. Bolton, R. Greiner and D.S. Wishart, ClassyFire: automated chemical classification with a comprehensive, computable taxonomy, *Journal of Cheminformatics* **8**(1) (2016), 61. doi:10.1186/s13321-016-0174-y.

[18] M. Kifer and H. Boley, RIF Overview (Second Edition), Technical Report, World Wide Web Consortium (W3C), 2013. https://www.w3.org/TR/rif-overview/.

[19] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean et al., SWRL: A semantic web rule language combining OWL and RuleML, *W3C Member Submission* **21**(79) (2004), 1–31.

[20] H. Boley, S. Tabet and G. Wagner, Design Rationale for RuleML: A Markup Language for Semantic Web Rules, in: *SWWS*, Vol. 1, 2001, pp. 381–401.

[21] B. Motik, U. Sattler and R. Studer, Query Answering for OWL-DL with Rules, *Journal of Web Semantics* **3**(1) (2005), 41–60.

[22] P.F. Patel-Schneider, A Proposal for a SWRL Extension Towards First-order Logic, *W3C Member Submission, April* (2005).

[23] T. Mossakowski, M. Codescu, F. Neuhaus and O. Kutz, The Distributed Ontology, Modeling and Specification Language–DOL, in: *The Road to Universal Logic*, Springer, 2015, pp. 489–520.

[24] O.M. Group, Distributed Ontology, Model, and Specification Language, Standard, Object Management Group, Milford, MA, USA, 2018.

[25] T. Mossakowski, C. Maeder and 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.

[26] T. Mossakowski and M. Codescu, The Heterogeneous Tool Set — some Recent Developments and Highlights, *24th International Workshop on Algebraic Development Techniques 2018* (2018).

[27] O. Kutz, J. Hastings and T. Mossakowski, Modelling highly symmetrical molecules: Linking ontologies and graphs, in: *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, Springer, 2012, pp. 103–111.

[28] M. Glauer, A. Memariani, F. Neuhaus, T. Mossakowski and J. Hastings, Interpretable Ontology Extension in Chemistry, *Semantic Web Journal* (accepted).

[29] J. Hastings, M. Glauer, A. Memariani, F. Neuhaus and T. Mossakowski, Learning chemistry: exploring the suitability of machine learning for the task of structure-based chemical ontology classification, *Journal of Cheminformatics* **13**(1) (2021), 1–20.

[30] A. Memariani, M. Glauer, F. Neuhaus, T. Mossakowski and J. Hastings, Automated and explainable ontology extension based on deep learning: A case study in the chemical domain, in: *International Workshop on Data meets Applied Ontologies in Explainable AI (DAO-XAI 2021)*, R. Confalonieri, O. Kutz and D. Calvanese, eds, CEUR Workshop Proceedings, Vol. 2998, http://ceur-ws.org/Vol-2998/, 2021. ISSN 1613-0073. http://ceur-ws.org/Vol-2998/paper1.pdf.