Automatically Extracting OWL Versions of FOL Ontologies

Torsten Hahmann(✉) and Robert W. Powell II

School of Computing and Information Sciences, University of Maine, Orono, ME 04469, USA
{torsten.hahmann,robert.powell}@maine.edu

Abstract. While OWL and RDF are by far the most popular logic-based languages for Semantic Web Ontologies, some well-designed ontologies are only available in languages with a much richer expressivity, such as first-order logic (FOL) or the ISO standard Common Logic. This inhibits reuse of these ontologies by the wider Semantic Web Community. While converting OWL ontologies to FOL is straightforward, the reverse problem of finding the closest OWL approximation of an FOL ontology is undecidable. However, for most practical purposes, a "good enough" OWL approximation need not be perfect to enable wider reuse by the Semantic Web Community.

This paper outlines such a conversion approach by first normalizing FOL sentences into a function-free prenex conjunctive normal (FF-PCNF) that strips away minor syntactic differences and then applying a pattern-based approach to identify common OWL axioms. It is tested on the over 2,000 FOL ontologies from the Common Logic Ontology Repository.

Keywords: Ontology translation · Common Logic · First-order logic · Web Ontology Language (OWL) · Prenex Normal Form (PNF)

1 Introduction

Ontologies make knowledge about our world explicit, with uses in a variety of settings, ranging from conceptual modeling and knowledge management, to the dissemination of the semantics of data on the web, and to automated reasoning that supports knowledge querying, discovery, and integration. Ontologies amendable to automated reasoning must be specified in a language with machine-interpretable formal semantics, such as various description logics including the Web Ontology Language, OWL2 [12,17], first-order logic or Common Logic [13], or rule languages like SWRL (https://www.w3.org/Submission/SWRL/). The specific choice of ontology language depends on a number of factors, including the complexity of the domain that is modeled, the amount of detail that needs to be expressed (including what kind of relations need to be modeled), the kind and

This material is based in part upon work supported by The National Science Foundation under grants OIA-1937099, OIA-2033607, and III-1565811.

A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 252–269, 2021.
https://doi.org/10.1007/978-3-030-88361-4_15

complexity of the reasoning that needs to be supported (e.g., verification of the ontology's internal consistency or its consistency with large data sets, querying of data, or only classification tasks), and the required reasoning efficiency. In the choice of language we make a trade-off between expressivity and tractability [4]. Description logics (see, e.g., [1]) sacrifice some expressivity for decidability or even tractability while first-order logic and more expressive languages sacrifice decidability for increased expressivity.

The OWL and OWL2 families of ontology languages [12,17] have become defacto standards for representing semantic knowledge to be used for lightweight reasoning such as classification tasks and consistency checking of a taxonomy. However, more expressive language, such as full first-order logic (FOL), are beneficial in settings when greater expressivity or flexibility in how knowledge is captured are paramount. For example, FOL permits use of functions and relations of arbitrary arity, which are critical for modeling spatio-temporal phenomena (which often add a temporal parameter to relations), and supports axiomatizing the interaction between relations in more detail. FOL has found a variety of uses, including for the specification of foundational ontologies such as DOLCE, BFO or GFO, for mid-level/generic ontologies (e.g., about spatial and/or temporal aspects or processes), and for domain reference ontologies such as the Hydro Foundational Ontology [11]. In many cases, the developed first-order ontologies primarily serve as reference representations (reference ontologies in the sense of [11,14]) that guide integration of ontologies across domains or help extract lightweight versions for specific purposes (e.g. DOLCE-Lite). But currently, these lightweight versions must be crafted by hand (see, e.g., [2]) which is not only costly but is further inhibited by many Semantic Web or domain experts being less familiar or less confident in working with FOL. Another issue with manually crafted OWL versions of FOL ontologies is the overhead of having to simultaneously maintain an OWL and a FOL version of an ontology and any potential discrepancies that may result. This motivates the work presented here: we want to develop an approach to automatically produce OWL versions from existing FOL ontologies. This will help leverage the significant resources that have already been invested in developing rigorous, densely axiomatized first-order logic ontologies and will make them accessible to a broader community of domain scientists who are more familiar with the OWL notation. It also would make the knowledge encoded in the FOL ontologies amendable to automated reasoning tasks that need to scale by magnitudes beyond what first-order reasoners currently can accomplish [20].

Because of the undecidability of FOL, computing a maximal OWL approximation of an FOL ontology is an intractable task that would require reasoning over its possibly infinite set of theorems. That is why instead of aiming for the elusive maximal approximation, we more pragmatically aim to efficiently produce "good enough" approximations. A "good enough" OWL2 ontology only needs to contain the kind of knowledge that an average OWL2 developer would have included in a hand-crafted, "native" OWL ontology, i.e. one that has been originally developed in OWL, for the same domain and scope.

2 Approach

Approximating a first order logic (FOL) ontology into a set of web ontology language (OWL) expressions presents multiple issues. The fact that the complexity of some FOL statements exceeds OWL's expressivity is not addressed here as it may require significant ontology re-engineering efforts. But a related issue is which OWL constructs to actually look for. This is addressed in Sect. 2.1 that identifies FOL templates of common OWL constructs. The additional issues of how to identify the portions of a FOL axiom that can be expressed via the available OWL expressions and how to deal with FOL's syntactic flexibility in encoding the same semantic content are tackled in Sect. 2.2, which develops a suitable normal form as basis for comparing FOL sentences against the templates. But even after normalization, matching of FOL sentences against the templates is rather expensive as discussed in Sect. 2.4. We develop a first-pass filtering approach described in Sect. 2.3. Figure 1 outlines our overall approach.

2.1 Common OWL Constructs as FOL Sentences

FOL provides a very small and generic set of logical connectives, but does not prescribe or constrain how to logically capture the semantic relationships between a set of non-logical symbols (i.e., the vocabulary of the domain) [11]. In contrast, OWL provides a large set of constructs, which are by design closely aligned with the kind of knowledge that people most commonly want to capture and which guide how to semantically relate a domain vocabulary. Consider as example the definition of the class `Father` from the OWL Primer [12]:

```
Father SubClassOf IntersectionOf(Man Parent)
```

In FOL, this could be expressed in multiple ways, for example[1]:

$$
\begin{aligned}
&\forall x[Father(x) \rightarrow Man(x) \wedge Parent(x)]\\
\Leftrightarrow &\forall x[\neg Man(x) \vee \neg Parent(x) \rightarrow \neg Father(x)]\\
\Leftrightarrow &\forall x[\neg Father(x) \vee (Man(x) \wedge Parent(x))]\\
\Leftrightarrow &\forall x[(\neg Father(x) \vee Man(x)) \wedge (\neg Father(x) \vee Parent(x))]\\
\Leftrightarrow &\neg\exists x[Father(x) \wedge (\neg Man(x) \vee \neg Parent(x))]\\
\Leftarrow &\forall x[Father(x) \leftrightarrow Man(x) \wedge Parent(x)]
\end{aligned}
$$

In comparison, OWL ontologies are less syntactically variable and heavily rely on simple cases of the available constructs. We mostly find taxonomic knowledge about classes and relations, domain and range restrictions on relations and classes, and simple properties of relations (reflexivity, symmetry, etc.) while more complex, nested class and property expressions are used sparingly even when permitted. A study of 518 OWL ontologies [7] has found that over 90% of class axioms are simple, meaning that they contain at most three class or property names. This observation informs our approach. It suggests starting with the

[1] The last sentence is not logically equivalent but still contains the same subclass relationship as one direction of the biconditional.

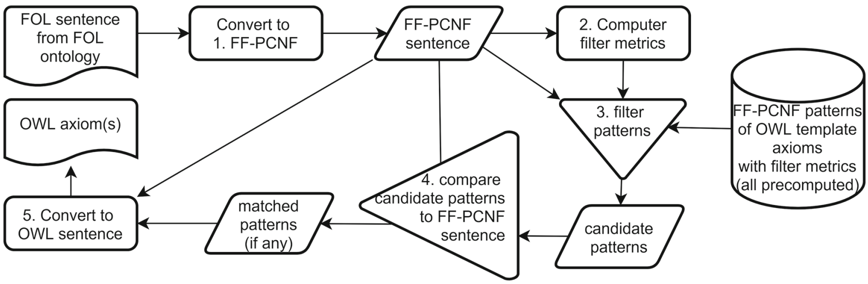

Fig. 1. Overview of our approach: Sentences are read from a FOL ontology and then converted to FF-PCNF (1). Once converted, metrics are computed (2) to filter the FF-PCNF for candidate templates (3) based on pre-computed template metrics. The sentences are then tested for exact matches against the templates' FF-PCNF (4). The matching ones produce OWL axioms (5).

OWL constructs and translating them to FOL rather than building an inventory of all the possible ways one could encode an OWL construct in FOL.

As summarized in Table 1, all Class and Object Property Axioms[2] from the OWL 2 Structural Specification [17] are supported[3]. But Class and Property Expressions therein are restricted as follows:

- Class expressions can make use of at most one propositional connective/-operation (`IntersectionOf`, `UnionOf`, or `ComplementOf`). Because our example definition of `Father` contains only one connective (an intersection) in its superclass expression, it is translated.
- Object property expressions may make use of one `InverseOf` expression.
- The Object Property Restrictions Existential Quantification, Universal Quantification, and Self-Restriction are supported but class expressions therein are also limited to a maximum of one propositional connective.
- Object Property Cardinality Restrictions are not supported as they are cumbersome to express in FOL and rarely, if at all, used in FOL.
- Class expressions involving Individuals (namely Enumeration of Individuals or Individual Value Restrictions) are not supported because individuals are not commonly included in FOL ontologies[4].

These templates still cover 97.4% of the simple class axioms from [7] and many additional role and complex class axioms not further broken down in [7]. Moreover, our restrictions are of no consequence for those OWL2 profiles that impose more stringent limits in the use of propositional connectives and inverses.

2.2 FF-PCNF for Dealing with Syntactic Variations in FOL

A specific challenge we have to overcome is that FOL is much less syntactically restricted than OWL as demonstrated by the Father construct. In order to

[2] Data Properties are indistinguishable from Object Properties in FOL and not used.
[3] Their exact FOL encoding does not really matter after the normalization step.
[4] All Individuals encountered during parsing are declared as such in the OWL output.

Table 1. OWL constructs with equivalent FOL sentences and FF-PCNF sentences that serve as templates for filtering and matching.

	OWL	Representative FOL sentence	FF-PCNF template
Class expression axioms			
1	`SubClassOf(C D)`	$\forall x[C(x) \rightarrow D(x)]$	$\forall x[\neg C(x) \vee D(x)]$
1a	`EquivalentClasses(C D)`	Inferred from SubClassOf axioms	
2	`DisjointClasses(C D)`	$\forall x[C(x) \rightarrow \neg D(x)]$	$\forall x[\neg C(x) \vee \neg D(x)]$
2a	`DisjointUnionOf(C D E ...)`	Inferred from SubClassOf and DisjointClasses axioms	
Object property axioms			
3	`SubObjectPropertyOf(R S)`	$\forall x, y[R(x,y) \rightarrow S(x,y)]$	$\forall x, y[\neg R(x,y) \vee S(x,y)]$
3a	`EquivalentObjectProperties(R S)`	Inferred from SubObjectPropertyOf axioms	
3b	`InverseObjectProperties(R S)`	inferred from SubObjectPropertyOf axioms (involving an inverse)	
4	`DisjointObjectProperties(R S)`	$\forall x, y[R(x,y) \rightarrow \neg S(x,y)]$	$\forall x, y[\neg R(x,y) \vee \neg S(x,y)]$
5	`ObjectPropertyDomain(R C)`	$\forall x, y[R(x,y) \rightarrow C(x)]$	$\forall x, y[\neg R(x,y) \vee C(x)]$
6	`ObjectPropertyRange(R C)`	$\forall x, y[R(x,y) \rightarrow C(y)]$	$\forall x, y[\neg R(x,y) \vee C(y)]$
7	`ReflexiveObjectProperty(R)`	$\forall x[R(x,x)]$	$\forall x[R(x,x)]$
8	`IrreflexiveObjectProperty(R)`	$\forall x[\neg R(x,x)]$	$\forall x[\neg R(x,x)]$
9	`SymmetricObjectProperty(R)`	$\forall x, y[R(x,y) \rightarrow R(y,x)]$	$\forall x, y[\neg R(x,y) \vee R(y,x)]$
10	`AsymmetricObjectProperty(R)`	$\forall x, y[R(x,y) \rightarrow \neg R(y,x)]$	$\forall x, y[\neg R(x,y) \vee \neg R(y,x)]$
11	`TransitiveObjectProperty(R)`	$\forall x, y, z[R(x,y) \rightarrow [R(y,z) \rightarrow R(x,z)]]$	$\forall x, y, z[\neg R(x,y) \vee \neg R(y,z) \vee R(x,z)]$
12	`FunctionalObjectProperty(R)`	$\forall x, y, z[R(x,y) \rightarrow [R(x,z) \rightarrow y = z]]$	$\forall x, y, z[\neg R(x,y) \vee \neg R(x,z) \vee = (y,z)]$
13	`InverseFunctionalObjectProperty(R)`	$\forall x, y, z[R(x,y) \rightarrow [R(z,y) \rightarrow x = z]]$	$\forall x, y, z[\neg R(x,y) \vee \neg R(z,y) \vee = (x,z)]$
Class axioms with existential or universal quantification or self-reference			
14	`SubClassOf(C ObjectSomeValuesFrom(R D))`	$\forall x \exists y[C(x) \rightarrow R(x,y) \wedge D(y)]$	$\forall x \exists y[[\neg C(x) \vee R(x,y)] \wedge [\neg C(x) \vee D(y))]]$
15	`SubClassOf(ObjectSomeValuesFrom(R D) C)`	$\forall x \exists y[R(x,y) \wedge D(y) \rightarrow C(x)]$	$\forall x \exists y[\neg R(x,y) \vee \neg D(x) \vee C(y)]$
16	`SubClassOf(C ObjectAllValuesFrom(R D))`	$\forall x, y[C(x) \wedge R(x,y) \rightarrow D(y)]$	$\forall x, y[\neg C(x) \vee \neg R(x,y) \vee D(y)]$
17	`SubClassOf(ObjectAllValuesFrom(R D) C)`	$\forall x, y[R(x,y) \rightarrow D(y) \rightarrow C(x)]$	$\forall x, y[[R(x,y) \vee C(x)] \wedge [\neg D(y) \vee C(x)]]$
18	`SubClassOf(C ObjectHasSelf(R))`	$\forall x[C(x) \rightarrow R(x,x)]$	$\forall x[\neg C(x) \vee R(x,x)]$
19	`SubClassOf(ObjectHasSelf(R) C)`	$\forall x[R(x,x) \rightarrow C(x)]$	$\forall x[\neg R(x,x) \vee C(x)]$

$$\forall x[A(x) \rightarrow \exists y[\neg(B(x,y) \vee \neg D(y))]] \tag{1a}$$
$$\equiv \forall x[\neg A(x) \vee \exists y[\neg(B(x,y) \vee \neg D(y))]] \tag{1b}$$
$$\equiv \forall x[\neg A(x) \vee \exists y[B(x,y) \wedge D(y)]] \tag{1c}$$
$$\equiv \forall x \exists y[\neg A(x) \vee (B(x,y) \wedge D(y))] \tag{1d}$$
$$\equiv \forall x \exists y[(\neg A(x) \vee B(x,y)) \wedge (\neg A(x) \vee D(y))] \tag{1e: FF-PCNF}$$
$$\equiv \forall x \exists y[\neg A(x) \vee (B(x,y) \wedge D(y)))] \tag{PNF; same as 1d}$$
$$\approx \forall x[(\neg A(x) \vee B(x,f(x)) \wedge (\neg A(x) \vee D(y))] \tag{CNF}$$

Fig. 2. Conversion of an example FOL sentence into FF-PCNF. The PNF and CNF conversions are included for comparison as the last two lines. Sentence (d) is where the prenex is formed and (e) is the result of distributing disjunctions over conjunctive terms. The final sentence (e) matches the FF-PCNF template 14.

identify certain OWL constructs, we have to manage this syntactic flexibility. We will do so using a normal form. Normal forms constrain the structure of an expression to enable more streamlined sentence processing for automated reasoning tasks. A normal form for easily comparing FOL expressions to the OWL constructs in Table 1 must fulfill three requirements: 1) make it easy to compare entire FOL sentences or portions thereof to the OWL constructs, 2) maintain existential quantification in order to identify `ObjectSomeValuesFrom` expressions, and 3) remove any function symbols.

Conjunctive normal form (CNF) is probably the most widely used normal form in knowledge representation. It represents a FOL sentence as a universally quantified sentence comprised of a single conjunction over several disjunctive terms. Such conjunctions over disjunctive terms are attractive for our purposes because the FOL versions of our OWL templates, with the exceptions of 14 and 17, only contain disjunctions. Thus by breaking a sentence into one big universally quantified conjunction over a set of disjunctions (the latter are commonly called "clauses") we can check each disjunction individually against the OWL templates. However, conversion to CNF (see, e.g., [4]) removes existential quantifiers during the Skolemization step and renders standard CNF unsuitable for our purposes. Prenex normal forms (PNF), on the other hand, maintain existential quantification by pulling out all quantifiers to the very front of the sentence, called the prenex (e.g. the $\forall x \exists y$ portion in Fig. 2(e)), followed by a quantifier-free portion called the matrix (e.g. the $(\neg A(x) \vee B(x,y)) \wedge (\neg A(x) \vee D(y))$ portion). To meet our needs, we alter the standard CNF conversion by replacing the Skolemization step by a prenex-forming step that moves both universal and existential quantifiers to the front. During this step, quantifiers are also heuristically coalesced to reduce the overall number of quantifiers as explained further down. Because OWL does not know function symbols, we re-encode them as predicates in a step before prenex construction. As explained further down, we substitute function symbols of arity n by new (n+1)-ary predicates with a new existentially quantified variable. Because of how we combine aspects of CNF and PNF and remove all functions, we call the result *function-free prenex conjunctive normal form* (FF-PCNF). Note that just like in the standard conversion to CNF, the final distribution step may exponentially increase the length and, thus, the number of FF-PCNF sentences because universally quantified conjunctions form separate sentences for the subsequent steps.

Function Substitution. Within the matrix, all n-ary functions are substituted by new (n+1)-ary predicates. Any occurrence of the function symbol in an atom is replaced by a conjunction over two terms: (1) the old atom with the functional term substituted by a new universally quantified variable and (2) the new (n+1)-ary predicate over the function's nested terms and the newly introduced variable. To maintain satisfiability two new sentences need to be added to ensure that the new (n+1)-ary predicates are indeed functional in their behaviour: (a) $\forall \overrightarrow{x} \exists y P_f(\overrightarrow{x}, y)$ (there is some result for every combination of inputs of the function) and (b) $\forall \overrightarrow{x}, y, z[P_f(\overrightarrow{x}, y) \land P_f(\overrightarrow{x}, z) \rightarrow y = z]$ (there is at most one result for any combination of inputs of the function). Note that these sentences do not need to be explicitly added to our FF-PCNF sentences; instead we can immediately add a `FunctionalObjectProperty` axiom on the newly introduced predicate P_f. Note further that function removal only yields OWL axioms for unary functions, because all other result in predicates of arity three or greater that are currently not converted to OWL.

Quantifier Coalescing. During prenex creation, there is an opportunity to shorten the final prenexes. Depending on variable placement in the sentence, like quantifiers and their variabes can be merged ("coalesced") into a single quantified variable, which will increase the chances that a sentence matches one of the FF-PCNF templates later on. Quantifier coalescing applies standard logical rules:

$$\forall x[A(x)] \land \forall y[B(y)] \iff \forall z[A(z) \land B(z)]$$

$$\exists x[A(x)] \lor \exists y[B(y)] \iff \exists z[A(z) \lor B(z)]$$

To leverage this potential without sacrificing efficiency we apply a greedy heuristic with a single look-ahead when deciding which quantifier to coalesce when there are multiple choices. If the parent term is a conjunction then universal quantifiers are coalesced, otherwise existential quantifiers are coalesced. In the case where the parent is a quantifier itself, it absorbs children with like quantifiers before applying the look-ahead for the merged quantifier again.

2.3 Filtering FF-PCNF Sentences by Templates

To utilize FF-PCNF as a normal form to identify the presence of OWL axioms within FOL axioms, we have converted the "representative" FOL translation of each OWL axiom templates from Table 1 into an FF-PCNF template as shown in column 4. To identify whether a particular sentence from an FOL ontology contains any OWL axioms, we convert the FOL sentence to FF-PCNF (step 1 in our approach) and then need an efficient way to compare the result against the stored FF-PCNF templates (step 3). This comparison can be extremely expensive: It is not a simple string comparison because of the variations in variable

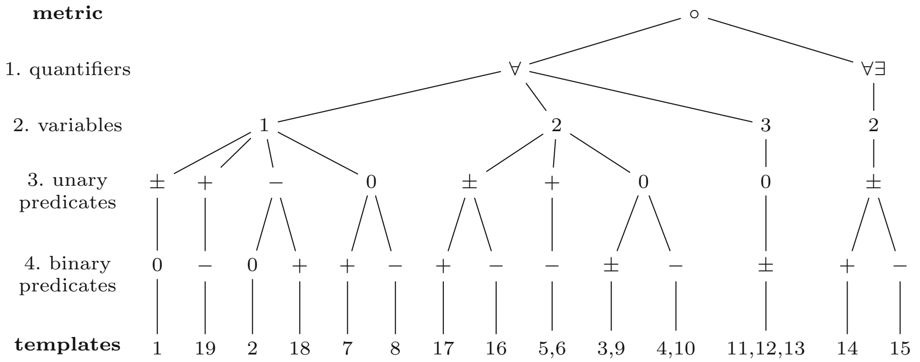

Fig. 3. Decision tree (implemented as set filters) of the four filtering metrics to identify potential OWL axioms in a FF-PCNF sentence. The leafs are the candidate templates from Table 1. Criteria 3 and 4 distinguish between presence or absence of positive and/or negative unary (criteria 3) and binary (criteria 4) predicates. In criteria 3 a "0" indicates the absence of unary predicates, + and −, respectively, the presence of only positive or only negated unary predicates, and ± the presence of both positive and negated unary predicates. Analogously for binary predicates in criteria 4. For example, template 16 requires two universally quantified variables, both positive and negated unary predicates (at least one of each), and only negated binary predicates. Combinations not in the decision tree lead to outright rejection of a sentence.

and predicate names or variations in the number of predicates. For example, the class $D(x)$ in template 1 could be named differently or be any anonymous union of classes. To keep the number of costly comparisons to a minimum, a filtering method first eliminates all obvious mismatches. Four efficiently computable filtering metrics (step 3) are suggested by the templates' syntactic structure:

1. the type of quantifiers found in the prenex,
2. the number of quantified variables,
3. the presence of unary or binary predicates in the matrix,
4. and the signs of the predicates (positive, negated or a mix).

Applied in that order, these metrics create the fourteen groups of sentences shown in Fig. 3, with ten groups leading to a unique template. Thus, in many cases a FF-PCNF sentence needs to be compared against only one template.

As a final filtering step, the number of atoms (i.e. the number of predicate instances) and the number of distinct predicates are used to further reduce the number of clauses that must be inspected closer for a match against some of the object property templates. Reflexive, irreflexive, symmetric, asymmetric, or transitive axioms can only be present if exactly one distinct predicate name is used. The number of atoms is also fixed: 1 for a reflexive or irreflexive property, 2 for a symmetric or asymmetric one, and 3 for a transitive one. Likewise, a functional or inverse functional axiom requires exactly two distinct predicate names, one of which must be the equals predicate.

2.4 Matching FF-PCNF Sentences Against Candidate Templates

The filtering drastically reduces the number of candidate FF-PCNF sentences – eliminating many altogether – that must be compared more closely against one or multiple candidate templates (step 4 in our approach). This comparison – the most expensive step of the conversion algorithm – then tests whether a FF-PCNF sentence precisely matches a candidate template. It typically involves checking variable use and placement across atoms within the clause. For example, the `ObjectPropertyDomain(R C)` and `ObjectPropertyRange(R C)` templates (5 and 6) only differ in where the variable in the unary predicate appears in the binary predicate. As another example, consider the sentences $\forall xy[\neg R(x,y) \vee S(x,y)]$ and $\forall xy[\neg R(x,y) \vee S(y,x)]$. By the filter metrics both match the templates for `SubObjectProperty(R S)`: they have two universally quantified variables and no unary predicates and a mix of positive and negated binary predicates. Thus filtering leaves templates 3 and 9 as candidates, but 9 is later ruled out because it is restricted to a single named predicate. Subsequently, the first sentence can be matched to the template. The second sentence, however, would not yet be a precise match because the variables in the predicate S are inverted. To create a precise match, the `InverseOf` needs to be added to the predicate S, resulting in the OWL axiom `SubObjectProperty(R InverseOf(S))`. When no match is established the FF-PCNF sentence is discarded.

2.5 Ensuring Adherence to OWL2 Global Restrictions

To guarantee decidability, OWL2 makes some global restrictions on the use of properties. Two restrictions on object properties are relevant to our translations. (1) The *simple role restriction* disallows use of complex object properties (roles) in constructs such as `FunctionalObjectProperty` or `DisjointObjectProperties`. To enforce it, we track all properties that are used in such constructs. At the end, we discard all axioms that would make these properties non-simply, namely transitive declarations (template 11) and axioms that use them within an `ObjectPropertyChain` construct. (2) Violations of the *property hierarchy restriction* only occur in the presence of multiple `ObjectPropertyChains` involving the same property. But these are quite rare in our translations: Only seven ontologies in our test set contain two or more `ObjectPropertyChains` and only one[5] actually violates the restriction. Thus, we defer to the `OWL API profile checker` tool[6] to identify such violations after producing OWL2 files and leave it up to human experts to resolve non-compliance.

Finally, we also allow choosing a target *OWL2 profile* [21]: Full (default), DL, EL, QL, or RL. To achieve this, disallowed object property axioms (e.g. FunctionalObjectProperty in EL and QL) and axioms wherein certain complex expressions are disallowed (e.g. `InverseOf` in EL; or `UnionOf` inside domain or range restriction axioms in EL, QL or RL) are discarded at the end.

[5] http://colore.oor.net/bipartite_incidence/owl/interval_incidence.all.owl.
[6] https://github.com/stain/profilechecker.

3 Implementation

The approach is implemented in Python 3 as part of the open-source project `macleod`[7]. The implementation utilizes an internal object structure to encode a FOL ontology, a parser to construct the internal object structure from CLIF files, methods for each type of object that support conversion into FF-PCNF, and methods for writing OWL axioms.

The internal object structure (see `src/macleod/logical/`) represents an ontology as a tree, each node encoding a logical or non-logical entity from a FOL ontology. Logical objects are: `Ontology`, sentences (`Axiom`), quantified formula (`Quantifier` with specializations `Existential` and `Universal`), connective formulas (`Connective` with specializations `Disjunction` and `Conjunction`), and negated formula (`Negation`). Atoms are represented as `Predicate`s and may contain functional terms, denoted as `Functions`. The various object types provide methods that support conversion to FF-PCNF. E.g., a `Negation` supports pushing negation inwards, a `Function` supports rewriting as a `Predicate`, and a `Conjunctive` supports distribution of disjunctions over conjunctions.

The parser (`src/macleod/parsing/parser.py`) utilizes a Backus-Naur grammar of a portion of the CLIF notation of Common Logic [13]. A lexer (an advanced tokenizer) and parser are built using Python's PLY library[8] to implement the grammar, to tokenize the CLIF files, and to finally parse them. Parsing substitutes implications and biconditionals by CNF sentences. It results in representing each CLIF file as an `Ontology` object, which contains the axioms and keeps track of all imported CLIF files, which are recursively parsed into separate `Ontology` objects. During parsing, additional information, such as lists of all predicate and function symbols and their arities, and variables names are saved for each `Ontology` and each `Axiom` for later use.

Python's ElementTree XML API[9] is used to write the axioms in OWL/XML format. For completeness, declaration axioms for all encountered predicates of arity two or less, i.e. all classes and object properties, are automatically included regardless of whether they appear in any resulting OWL axiom.

4 Experimental Results

4.1 Materials

We have tested the approach on ontologies from the Common Logic Ontology Repository (COLORE: http://colore.oor.net), which currently contains over 2,700 files with sentences in the CLIF syntax of the Common Logic standard [13]. Some do not specify ontologies per se, but rather theorems, mappings between ontologies, partial models, or serve archival purposes. Of the 2,283 files that do represent ontologies or modules thereof (like individual definitions), 2,102

[7] https://github.com/thahmann/macleod.

[8] PLY is a Python port of the standard Unix tools Lex and Yacc.

[9] https://docs.python.org/3/library/xml.etree.elementtree.html; the Owlready2 module was another option but writing axioms was not as straightforward.

(92%) were successfully parsed; others either contain syntax errors or make use of unsupported Common Logic constructs that go beyond standard FOL. Our first evaluation uses all FOL sentences from the 2,102 successfully parsed files. Our second evaluation uses entire ontologies – i.e. CLIF files recursively closed under the `cl:imports` construct – rather than individual files. For 1,965 ontologies all imported modules can be parsed correctly. Of those, we select the 302 that contain a minimum of 15 predicates (unary or binary ones) and 15 axioms. Many smaller ontologies do not meet the predicate threshold; they primarily serve as modules of larger ontologies or are theories of common mathematical structures used as tools for verifying other ontologies. The 302 utilized ontologies range from 15 to 128 unary and binary predicates (median of 24) and 18 to 246 axioms (median of 69). While these may still be small compared to OWL ontologies, they are quite sizable for FOL ontologies.

To avoid distorting our results by many fairly similar ontologies, we group them by hierarchy. A hierarchy shares a signature and often a substantial set of imports (and, thus, axioms) [10]. The utilized ontologies span 33 hierarchies, 11 of which reside in a hierarchy of their own (listed first in Fig. 4)[10]. Of the remaining 22 hierarchies (bottom of Fig. 4) 17 contain 2 to 11 ontologies, and five are larger hierarchies with 20 to 76 ontologies.

4.2 Results

All tests are conducted using Python 3.7 on a Windows 10 laptop (i5-8350, 4 cores at 1.7 GHz base frequency, 8 GB RAM). The reported times are wall times that include parsing the CLIF file and its import closure.

The first experiment translates all 2,102 parseable CLIF files individually[11]. Altogether, they contain 4,257 FOL sentences, but only 3,387 (78%) of them use only predicates of arity one or two and can reasonably be expected to yield translations. They yielded 7,941 FF-PCNF sentences[12]. Filtering identified 5,957 FF-PCNF sentence-template pairs (on average 0.75 per FF-PCNF sentence and 1.76 per FOL sentence). 2,241 of these candidates produced OWL axioms, which amounts on average to 0.66 OWL axioms per FOL sentence. The whole experiment (including parsing, filtering and matching) finished in 151s apart from one ontology, namely `periods/periods_over_rationals.clif`, that increased exponentially in length and whose conversion and filtering/matching took 265s alone but did not yield any OWL axioms. Table 2 summarizes the axiom distribution: they are almost equally divided between class and object property axioms. 28.3% are subclass axioms, with the majority (540; 24.1% of total) being simple while 95 (4.2% of total) are complex. Among object property axioms, domain and range restrictions (18.5%) are most prevalent, followed by subproperty axioms

[10] For the `gwml2` and the `simple_features` we only work with the complete ontologies because the submodules are not particularly meaningful on their own.

[11] Full results are available from https://github.com/thahmann/macleod/blob/master/research/ISWC2021-experimental-data.xlsx and the OWL2 outputs are provided in https://colore.oor.net/ in the `owl` subfolder of each ontology hierarchy.

[12] Recall that universally quantified conjunctions are split into separate sentences.

Table 2. Summary of the OWL axioms obtained from all parseable CLIF modules.

FOL axioms		**FF-PCNF**	**candidate**	**# OWL2**	**Prop. Class**	**Inverses**	**Property**
total	**arity** ≤ 2	**sentences**	**templates**	**Axioms**	**Operations**		**Chains**
4,257	3,387	7,941	5,957	2,241	236	158	30

total #	**Class Axioms:** *53%*			**ObjectProperty Axioms:** *47%*			
OWL2 Axioms	**SubClass**	**SomeValuesFrom/ AllValuesFrom**	**Disjoint Classes**	**Sub Properties**	**Disjoint Property**	**Domain/ Range R.**	**Other**
2,241	635 *28.3%*	310 *13.8%*	194 *8.7%*	249 *11.1%*	61 *2.7%*	414 *18.5%*	336 *15.0%*

(11.1%). The remainder are property disjointness (2.7%) and various property descriptors such as (ir)reflexivity, (a)symmetry, or functional properties. Even with the imposed limitation on unions, intersections, and complements, we produced 236 such operations in class expressions. Inverses and property chains were used 158 and 30 times, respectively.

The results from our second experiment on 302 ontologies with at least 15 axioms and 15 predicates of arity ≤ 2 are summarized in Fig. 4. It took on average 3.6s to convert these ontologies, though with some larger ontologies taking a bit longer: `molecular_graph/definitions/most_elements.clif` with 246 axioms and 128 predicates took over 23s, while the larger ontologies in the `multidim_space_physcont` hierarchy also took up to 14s.

One measure of efficacy is the number of OWL axioms produced per FOL sentence: It ranges from 0.4 to 1.33 across hierarchies (with a max. of 2.42 for individual ontologies), though most fall within 0.73 ± 0.23 OWL axioms (median + standard deviation). However, this is a purely statistical measure and does not capture how much of the semantics are preserved: It neither measures how many FOL axioms are *fully* translated nor does it normalize by the length, density or complexity of the source FOL axioms. A better way to judge the quality of the produced ontologies is by comparing them to "native" OWL2 ontologies. One established criteria for comparing the quality of ontologies is their semantic richness (or "axiom density") [8,19] that captures how tightly classes are constrained. It is typically measured in terms of the axiom-class ratio, for which we obtain a median of 4.00 across hierarchies. But the FOL ontologies in our experiments contain more properties (a median of 14.3) than classes (median of 11.0) which is not typical for OWL ontologies[13]. Thus, an axiom-concept ratio that divides the number of axioms by the total number of classes *and* properties is a more appropriate metric. We obtain a mean of 1.62 across the hierarchies, though with a fairly wide spread (0.40 to 2.21). Nevertheless, all but 3 hierarchies (`location_varzi`, `vision_cardworld`, `financial`) have an average ratio of one or more axioms per concepts.

5 Discussion

The results demonstrate that our approach is able to quickly extract OWL2 versions even from sizable FOL ontologies. It is expected to scale well because

[13] The 514 ontologies in [7] contain 618,260 classes but only 22,046 properties.

	# Ont.	# FOL sentences	# FOL sentences of arity ≤2	# Predicates of arity ≤2 (unary/binary) - higher arity predicates	PCNF sentences	candidate patterns	# of OWL axioms	OWL axioms per FOL sentence	axioms per concept (i.e. predicates of arity ≤2)	axiom per class	subclass	disjoint classes	someValuesFrom/ allValuesFrom	subproperty	disjoint properties	domain/ range restrictions	other property axioms	Average Conversion Time in s
\simple_features\sfc_fol.clif		110	110	59 (35/24) - 0	307	292	**130**	**1.18**	**2.20**	3.71	51	15	6	24	7	19	8	**4.70**
\mapsee\mapsee.clif		39	39	25 (13/12) - 0	74	112	**52**	**1.33**	**2.08**	4.00	16	6	7	0	0	23	0	**0.79**
\ciro\ciro-core.clif		115	81	41 (20/21) - 11	229	192	**81**	**1.00**	**1.98**	4.05	22	6	17	3	0	25	8	**5.29**
\chains_process\chains_state.clif		54	46	21 (9/12) -3	111	116	**40**	**0.87**	**1.90**	4.44	4	6	5	1	0	18	6	**1.58**
\cyclic_process\cyclic_state.clif		51	43	21 (9/12) -3	101	116	**40**	**0.93**	**1.90**	4.44	4	6	5	1	0	18	6	**1.42**
\process_specification_language\psl_outercore.clif		104	72	31 (11/20) - 11	199	151	**52**	**0.72**	**1.68**	4.73	8	6	5	3	0	23	7	**3.77**
\combined_time\combined_time_def.clif		22	20	15 (4/11) -2	73	142	**23**	**1.15**	**1.53**	5.75	0	1	4	0	0	16	2	**2.40**
\gwml2\gwml2_full.clif		81	81	66 (49/17) - 0	136	189	**88**	**1.09**	**1.33**	1.80	52	14	14	0	0	3	5	**3.57**
\velocity\speed.clif		56	56	26 (11/15) - 0	106	95	**32**	**0.57**	**1.23**	2.91	2	6	1	10	3	4	6	**1.06**
\molecular_graph\definitions\most_elements.clif		246	246	128 (126/2) - 0	359	143	**129**	**0.52**	**1.01**	1.02	129	0	0	0	0	0	0	**23.47**
\location_varzi\region_location.clif		23	20	20 (3/17) - 2	65	37	**8**	**0.40**	**0.40**	2.67	0	0	2	3	1	0	2	**0.89**
the following numbers (except the number of ontologies) are all averages over all ontologies in each hierarchy that have 15 or more axioms and predicates																		
\multidim_space_space\ & \multidim_space_spch	5	102	95	33 (15/18) - 1	202	198	**73**	**0.78**	**2.21**	4.82	14.0	9.0	4.0	16.0	4.0	18.0	9.0	**9.01**
\multidim_space_voids	8	155	146	46 (21/25) - 1	296	401	**101**	**0.70**	**2.19**	4.76	22.0	9.0	6.5	22.3	4.3	23.8	13.4	**6.70**
\multidim_space_physcont	32	185	176	59 (21/37) - 1	351	503	**127**	**0.72**	**2.18**	5.91	22.0	9.0	7.3	33.6	5.6	32.4	16.7	**9.92**
\multidim_space_codi(b)	59	57	57	59 (21/37) - 1	133	102	**46**	**0.83**	**1.86**	5.26	9.7	4.7	1.9	14.7	3.2	6.7	5.4	**1.75**
\direct_quality_process	2	57	49	23 (10/13) - 3	164	151	**41**	**0.84**	**1.83**	4.14	4.0	6.0	5.0	1.0	0.0	18.0	7.0	**2.95**
\injection_bipartite_process	2	57	49	23 (10/13) - 3	164	151	**41**	**0.84**	**1.83**	4.14	4.0	6.0	5.0	1.0	0.0	18.0	7.0	**3.35**
\onto_stit	2	41	37	19 (6/14) - 1	77	75	**35**	**0.93**	**1.81**	6.27	0.0	11.5	5.0	4.0	0.0	11.0	3.0	**0.74**
\weak_bipartite_process	2	55	47	23 (11/13) - 3	116	134	**41**	**0.88**	**1.80**	3.99	4.0	6.0	5.0	1.0	0.0	18.0	7.0	**1.47**
\multidim_occupy	5	120	96	37 (17/20) - 4	236	149	**66**	**0.70**	**1.78**	3.91	6.0	16.6	13.4	10.0	3.0	7.2	9.6	**2.43**
\mcg_process	2	58	50	24 (11/13) - 3	124	137	**41**	**0.82**	**1.76**	3.85	4.0	6.0	5.0	1.0	0.0	18.0	7.0	**1.89**
\rcc_continuous_process	4	76	68	24 (11/13) -3	286	157	**41**	**0.60**	**1.74**	3.85	4.0	6.0	5.0	1.0	0.0	18.0	7.0	**7.27**
\dolce\ and \dolce_XYZ	20	72	64	47 (39/8) -3	145	123	**77**	**1.21**	**1.71**	2.03	48.0	25.0	3.5	0.0	0.0	0.5	0.0	**1.77**
\multidim_mereotopology_codi(b)	76	67	67	23 (8/16) - 0	128	108	**40**	**0.61**	**1.68**	5.59	3.3	4.4	0.8	13.5	3.6	5.5	8.5	**1.66**
\most	3	59	36	19 (13/5) - 8	207	96	**34**	**0.88**	**1.64**	2.40	9.5	6.0	6.5	2.0	2.0	8.0	0.0	**4.13**
\size	6	92	79	33 (15/18) - 3	181	155	**53**	**0.66**	**1.59**	3.76	4.0	11.5	10.0	10.0	3.0	6.0	8.5	**2.46**
\psl_XYZ	11	64	53	25 (10/14) - 4	110	106	**38**	**0.72**	**1.54**	3.68	6.8	5.7	6.7	1.7	0.0	11.0	5.8	**1.57**
\tripartite_incidence	29	35	32	18 (8/10) - 3	97	53	**27**	**0.86**	**1.51**	3.41	5.0	3.0	2.3	2.0	2.0	12.0	1.0	**1.46**
\multidim_mereotopology_omt(b)	8	79	69	23 (8/15) - 2	158	97	**35**	**0.50**	**1.50**	4.46	3.4	4.0	0.4	12.0	3.3	4.5	7.3	**2.03**
\owltime\ and \owltime_XYZ	6	34	32	16 (5/11) -1	51	71	**25**	**0.84**	**1.47**	5.25	4.0	1.0	0.0	4.0	0.8	13.7	1.0	**0.50**
\bipartite_incidence	2	21	18	18 (5/13) - 4	62	41	**21**	**1.19**	**1.16**	4.20	3.0	1.0	0.0	6.0	2.0	8.0	1.0	**1.15**
\vision_cardworld	5	53	39	22 (6/16) - 6	128	80	**19**	**0.52**	**0.91**	3.14	4.0	3.0	3.0	1.6	0.8	6.0	1.0	**3.47**
\financial	2	99	99	91 (53/38) - 0	113	94	**57**	**0.56**	**0.61**	1.04	35.5	3.0	18.5	0.0	0.0	0.0	0.0	**1.33**
Totals:	302	2537	2270 (89.5%)		per FOL sentence				*mean % of total OWL axioms:*		*29%*	*13%*	*10%*	*12%*	*3%*	*24%*	*10%*	
MEAN		***77***	***69***	***35***	***2.6***	***2.3***	***53.1***	***0.82***	***1.62***	***3.92***	***15.4***	***6.8***	***5.5***	***6.2***	***1.5***	***12.5***	***5.3***	***3.57***
MEDIAN		***59***	***56***	***24***	***2.4***	***2.3***	***41.0***	***0.83***	***1.71***	***4.00***	***4.0***	***6.0***	***5.0***	***2.0***	***0.0***	***12.0***	***6.0***	***2.03***
MIN		*21*	*18*	*15*	*1.1*	*0.6*	*8.0*	*0.40*	*0.40*	*1.02*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.50*
MAX		*246*	*246*	*128*	*5.8*	*7.1*	*130.0*	*1.33*	*2.21*	*6.27*	*129.0*	*25.0*	*18.5*	*33.6*	*7.0*	*32.4*	*16.7*	*23.47*
STD DEVIATION		*47*	*46*	*24*	*0.9*	*1.0*	*31.2*	*0.23*	*0.43*	*1.27*	*24.7*	*5.1*	*4.5*	*8.1*	*1.9*	*8.3*	*4.0*	*4.21*

Fig. 4. Results from converting 302 ontologies from COLORE that contain at least 15 unary or binary predicates and at least 15 axioms.

the sentence by sentence conversion makes the time needed mainly dependent upon the number of candidates that need to be matched after filtering, which is linearly related to the number of FOL sentences.

The most critical evaluation aspect is the correctness of the resulting OWL2 ontologies. We have checked all 302 ontologies for syntactic correctness and conformance with OWL2-Full using the OWL API profile checker, while spot-checking adherence to more restricted OWL2 profiles when selected. The produced ontologies can also be successfully loaded in the Protege ontology development environment and be used for reasoning, such as classification, with off-the-shelf OWL2 reasoners such as Hermit.

To evaluate the quality of the produced ontologies, we primarily rely on the axiom-concept ratio as an indicator for their semantic richness in comparison to "native" OWL ontologies, which were originally developed in OWL. While our average axiom-concept ratio of 1.62 (over hierarchies) is lower than the average of 2.05 over the 518 native OWL ontologies (with over 1.7M axioms) from [7], our median of 1.71 is actually higher than theirs (1.62). That means more than half of our ontologies – which are essentially produced for free now – are already semantically richer than half of the existing OWL ontologies. The much lower variance (indicated by the standard deviation of 0.45) compared to that of 2.25 in [7] is evidence that we can consistently deliver OWL ontologies of high quality across domains – likely because of the higher quality of the FOL ontologies. With a few exceptions, such as `/location_varzi/region_location.clif` and the `/financial/` hierarchy, this can be taken as evidence that our OWL2 outputs are already "good enough" to be usable for many practical purposes.

The generated axioms also exhibit more diversity than the native OWL ontologies. The analyzed OWL ontologies in [7] consist of 55% simple subclass axioms (varying between 41 and 62% for different benchmark sets) and 24% subclass axioms with existential quantification (someValuesFrom), while property axioms make up only 5.2% (2.4% being domain and range restrictions). Not a single disjointness axiom was found among the native OWL ontologies. These numbers confirm the perception that native OWL ontologies often leave properties underdeveloped. The stark differences in use of property axioms (over 47% of all axioms in our results) underline that translating FOL ontologies can yield OWL ontologies that may often be richer – especially in the axiomatization of properties – than native OWL ontologies.

An initially *unanticipated* side benefit is the increase in intelligibility of FOL ontologies via translations. It provides developers of FOL ontologies access to a wealth of OWL development tools. Protege's (albeit) simple taxonomic and graphical visualizations of the resulting (inferred) class and property hierarchies, especially in combination with the integrated reasoners (e.g. Hermit), allowed us to spot axiomatization errors in FOL ontologies. Identifying these issues directly from the CLIF source was non-trivial because they were the result of axioms being combined across multiple CLIF files. With the help of the OWL reasoners' justifications and the log of the OWL axioms FOL sources, we could trace the errors to the originating FOL files and specific axioms.

Limitations. As initially discussed, an ontology's theory can be axiomatized in dramatically different ways, up to entirely disjoint sets of axioms [10]. This means that some knowledge that would be relevant to an OWL version may not be explicitly represented, but only inferred. Our template-based approach currently does not aim to infer such knowledge. It would require a *semantic translation* approach that can add to the OWL ontology by strategically or systematically guessing additional axioms (e.g. predicted subclass relationships or disjointness of sibling classes) that can be added after successful proving by an FOL theorem prover. Because of the intractability of FOL reasoning, such an approach will be limited in practice. But the potential benefits can be glimpsed at through one specific example:`/multidim_mereotopology_codi/codi_with_theorems.clif` is logically equivalent to `/multidim_mereotopology_codi/codi.clif` but explicitly adds (successfully proved) theorems that, for example, establish disjointness of properties. The difference in the outcome is striking: the number of OWL axioms increases from 32 to 52, raising the axiom-to-concept ratio from a mediocre 1.42 to 2.42, the highest among all translated ontologies and landing within the top quartile of native OWL2 ontologies.

6 Related Work

The idea of translating knowledge between different knowledge representation formalisms has been studied previously, for example in the Ontolingua [9], OntoMorph [5], and OntoMerge [6] systems and the distributed ontology language (DOL) [15], all of who aim to combine knowledge from ontologies represented in different languages. Ontolingua employs an intermediary language for which syntactic translations are defined to each knowledge representation language. OntoMorph employs direct syntactic translations between pairs of languages while also sketching the idea of semantic translations. OntoMerge also employs an internal language that is the result of syntactic translations of a source language, but then performs reasoning on the internal language before syntactically translating inferences. The DOL [15] provides a meta-language for specifying relationships between ontologies that are specified in different logical languages. However, reasoning with such heterogeneous sets of ontologies is expensive and intractable as it involves meta-reasoning over multiple logics. Moreover, as is the case with CLIF, reasoning support is limited. Currently, the heterogeneous toolset (HETS) [16] is the only tool that supports the DOL language and many available off-the-shelf reasoners for FOL and OWL cannot be reused. In contrast, our work on translation from FOL to OWL is more narrowly concerned with overcoming syntactic, semantic, and pragmatic differences between these two specific languages in order to make existing FOL more widely accessible and leverage the wider tool availability for OWL ontologies.

The theoretical basis of description logics [1] serve as foundations for bridging different ontologies languages, specifically propositional, description and first-order logic. Borgida [3] in particular provides formal translations to FOL for the syntactic constructs found in DL, the formal underpinning of OWL. These translations are leveraged here to express OWL axioms as semantically equivalent FOL sentences that serve as extraction templates.

The tool ROWLTab [18], also uses a PNF to translate from the rule-base language SWRL to OWL. But it differs in its overall goal, aiming to support domain experts in developing *new* OWL ontologies. We focus instead on creating OWL versions of *existing* FOL ontologies to increase accessibility and reuse. An example is the work by [2], who painstakingly translated a single ontology. We aim instead for less detailed but fast, cheap and fully automated translations.

7 Summary

Unrestricted usage of FOL results in an undecidable ontology [4] that effectively curbs the ontology's utilization where tractable reasoning is required. At the same time, the expressive capabilities of FOL, its flexibility, and its established formal underpinnings, still speak in favor of FOL as a representation language for reference ontologies. But existing FOL ontologies – which are the result of countless hours of ontology development and verification – are largely inaccessible to many knowledge engineers who are unfamiliar or uncomfortable with FOL. Moreover, there is a dearth of tools available to support the development, extension, or adoption of FOL ontologies. To widen the accessibility and usability of those FOL ontology, we have proposed a pragmatic ontology engineering approach to automatically extract OWL2 approximations from FOL ontologies that conform to specific desired OWL2 profiles. This essentially produces high-quality OWL2 ontologies for free now. These OWL ontologies can be inspected, extended, and used as the foundation for future development and can benefit from all available OWL tooling, such as for ontology visualization and evaluation. This helps to verify, evolve, and reuse the source FOL ontologies. More importantly, it avoids redundant ontology engineering efforts or maintaining copies of the ontologies in two languages with different expressivity (FOL and OWL).

We proposed FF-PCNF as an intermediate representation to more easily identify OWL patterns from FOL sentences despite FOL's syntactic flexibility. We demonstrated the practical usability and scalability of the approach by generating 2,241 OWL axioms from 3,387 FOL sentences in 150 s using a single core of a modern CPU and a negligible amount of memory. While the resulting ontologies make heaviest use of five OWL constructs (subclasses, domain and range restrictions, disjoint classes, subproperties), all 19 axiom templates are used to some extent.

Future Work needs to apply a broader set of ontology metrics (see e.g. [8,19]) to evaluate the produced ontologies and to identify better measures of the amount of semantics that are preserved by the translation. We further hope that our results can serve as baseline for continuous improvement of FOL-to-OWL translations. Potential avenues for improvement include tackling predicates of higher arities or inferring additional OWL axioms using FOL theorem proving.

Acknowledgment. We thank the four anonymous reviewers for their comments and suggestions that helped improve the final version.

References

1. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. IHIS, pp. 21–43. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92673-3_1
2. Benevides, A.B., Bourguet, J.R., Guizzardi, G., Peñaloza, R., Almeida, J.: Representing a reference foundational ontology of events in SROIQ. Appl. Ontol. **14**(3), 293–334 (2019). https://doi.org/10.3233/AO-190214
3. Borgida, A.: On the relative expressiveness of description logics and predicate logics. Artif. Intell. **82**(1–2), 353–367 (1996). https://doi.org/10.1016/0004-3702(96)00004-5
4. Brachman, R.J., Levesque, H.J.: Knowledge Representation and Reasoning. Elsevier (2004)
5. Chalupsky, H.: OntoMorph: a translation system for symbolic knowledge. In: KR 2000, pp. 471–482. Morgan Kaufmann (2000)
6. Dou, D., McDermott, D., Qi, P.: Ontology translation on the semantic web. In: Spaccapietra, S., Bertino, E., Jajodia, S., King, R., McLeod, D., Orlowska, M.E., Strous, L. (eds.) Journal on Data Semantics II. LNCS, vol. 3360, pp. 35–57. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30567-5_2
7. Eberhart, A., Shimizu, C., Chowdhury, S., Sarker, M.K., Hitzler, P.: Expressibility of OWL axioms with patterns. In: Verborgh, R., et al. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 230–245. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_14
8. García, J., García-Peñalvo, F.J., Therón, R.: A survey on ontology metrics. In: Lytras, M.D., Ordonez De Pablos, P., Ziderman, A., Roulstone, A., Maurer, H., Imber, J.B. (eds.) WSKS 2010. CCIS, vol. 111, pp. 22–27. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16318-0_4
9. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (1993)
10. Grüninger, M., Hahmann, T., Hashemi, A., Ong, D., Ozgovde, A.: Modular first-order ontologies via repositories. Appl. Ontol. **7**(2), 169–209 (2012). https://doi.org/10.3233/AO-2012-0106
11. Hahmann, T., Stephen, S.: Using a hydro-reference ontology to provide improved computer-interpretable semantics for the groundwater markup language (GWML2). Int. J. Geogr. Inf. Sci. **32**(6), 1138–1171 (2018). https://doi.org/10.1080/13658816.2018.1443751
12. Hitzler, P., Parsia, B., Patel-Schneider, P., Rudolph, S.: OWL 2 Web Ontology Language Primer (Second Edition) (2012). https://www.w3.org/TR/owl2-primer/
13. ISO 24707:2018 Common Logic (CL): a framework for a family of logic-based languages (2018). https://www.iso.org/standard/66249.html
14. Menzel, C.: Reference Ontologies - Application Ontologies: Either/Or or Both/And? In: WS on Reference and Application Ontologies at KI-03 (2003)
15. Mossakowski, T., Codescu, M., Neuhaus, F., Kutz, O.: The distributed ontology, modeling and specification language – DOL. In: Koslow, A., Buchsbaum, A. (eds.) The Road to Universal Logic. SUL, pp. 489–520. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15368-1_21

16. Mossakowski, T., Maeder, C., Lüttich, K.: The heterogeneous tool set, HETS. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 519–522. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71209-1_40
17. Motik, B., Patel-Schneider, P., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) (2012). https://www.w3.org/TR/owl2-syntax/
18. Sarker, M.K., Krisnadhi, A., Carral, D., Hitzler, P.: Rule-based OWL modeling with ROWLTab Protégé plugin. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10249, pp. 419–433. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58068-5_26
19. Sicilia, M., Rodríguez, D., García-Barriocanal, E., Sánchez-Alonso, S.: Empirical findings on ontology metrics. Expert Syst. Appl. **39**(8), 6706–6711 (2012). https://doi.org/10.1016/j.eswa.2011.11.094
20. Stephen, S., Hahmann, T.: Model-finding for externally verifying FOL ontologies: a study of spatial ontologies. In: International Conference on Formal Ontologies in Information Systems (FOIS 2020), pp. 233–248. IOS Press (2020). https://doi.org/10.3233/FAIA200675
21. Wu, Z., Fokoue, A., Grau, B., Horrocks, I., Motik, B.: OWL 2 Web Ontology Language Profiles (Second Edition) (2012). https://www.w3.org/TR/owl2-profiles/