Using Vampire to Reason with OWL

Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer and Ian Horrocks

Department of Computer Science
The University of Manchester
Manchester, UK
{tsarkov|riazanov|seanb|horrocks}@cs.man.ac.uk

Abstract. OWL DL corresponds to a Description Logic (DL) that is a fragment of classical first-order predicate logic (FOL). Therefore, the standard methods of automated reasoning for full FOL can potentially be used instead of dedicated DL reasoners to solve OWL DL reasoning tasks. In this paper we report on some experiments designed to explore the feasibility of using existing general-purpose FOL provers to reason with OWL DL. We also extend our approach to SWRL, a proposed rule language extension to OWL.

1 Introduction

It is well known that OWL DL corresponds to the $\mathcal{SHOIND}_n^-$ Description Logic (DL), and that, like most other DLs, $\mathcal{SHOIND}_n^-$ is a fragment of classical first-order predicate logic (FOL) [8, 17, 1]. This suggests the idea of using standard methods of automated reasoning for full FOL as a mechanism for reasoning with OWL DL, not necessarily with a view to replacing dedicated DL reasoners, but perhaps as a supplementary mechanism that can be used in development, testing and to attempt problems that are outside the expressive range of existing DL reasoners.

This might be done by trying to create from scratch new architectures for reasoning in full FOL, which would be specialised for dealing efficiently with typical DL reasoning tasks. A much less expensive option is to use existing implementations of FOL provers, with the possibility of making adjustments that exploit the structure of DL reasoning tasks. In this paper we investigate, experimentally, the viability of the latter approach. It should be noted that our current implementation is very simplistic, and is only intended as a preliminary feasibility study.

Our main experiment is organised as follows. We use a DL reasoner, FaCT++, to classify three reasonably realistic OWL ontologies (DL knowledge bases): Galen, Tambis and Wine. A large number of nontrivial class (concept) subsumption subtasks are extracted, translated into FOL and submitted to Vampire, a state-of-the-art general-purpose resolution-based FOL prover. This allows us to compare the performance of the two systems on the same problem set.

Although by no means suggesting that dedicated DL reasoners are redundant, the results of our experiment are quite encouraging, and show that Vampire can cope with the vast majority of problems derived from the classification of these ontologies. Moreover, our experiments show that some fairly simple pre-processing optimisations can greatly improve Vampire's performance on ontology derived problems, and suggest

that there may be scope for further significant improvements. At the same time, a significant number of tests were too hard for the FOL prover, which indicates that there may be a place for DL reasoning oriented optimisations in the Vampire architecture. Finally, a valuable byproduct of this experiment is a large set of realistic FOL problems which may be of interest to developers of other FOL systems.

In a second experiment, we use FOL translation and Vampire to solve problems from the OWL Test suite [25] designed to test reasoners for OWL DL. Although OWL DL corresponds to an expressive description logic for which the consistency problem is known to be decidable, there is as yet no effective decision procedure known for the complete language,[1] and consequently no implemented DL reasoners [18, 17]. Since no tableaux procedure is known for this DL, it is interesting to check if a general-purpose FO system can at least cope with some of the tasks in the OWL Test suite [25]. In fact it turns out that Vampire can solve the vast majority of these problems.

Finally, we show how the simplicity and flexibility of our FOL translation approach allows it to be trivially extended to deal with the Semantic Web Rule Language (SWRL), a proposed rule language extension to OWL [13].

2 Preliminaries

There have been earlier investigations of the use of FOL provers to reason with description logics. Paramasivam and Plaisted, for example, have investigated the use of FOL reasoning for DL classification [26], while Ganzinger and de Nivelle have developed decision procedures for the guarded fragment, a fragment of FOL that includes many description logics [10]. The most widely known work in this area was by Hustadt and Schmidt [20], who used the SPASS FOL prover to reason with propositional modal logics, and, via well known correspondences [30], with description logics. Their technique involved the use of a relatively complex functional translation which produces a subset of FOL for which SPASS can be tuned so as to guarantee complete reasoning. The results of this experiment were quite encouraging, with performance of the SPASS based system being comparable, in many cases, with that of state of the art DL reasoners. The tests, however, mainly concentrated on checking the satisfiability of (large) single modal logic formulae (equivalently, OWL class descriptions/DL concepts), rather than the more interesting task (in an ontology reasoning context) of checking the satisfiability of formulae w.r.t. a large theory (equivalently, an OWL ontology/DL knowledge base).

In all of the above techniques, the DL is translated into (the guarded fragment of) FOL in such a way that the prover can be used as a decision procedure for the logic—i.e., reasoning is sound, complete and terminating. Such techniques have, however, yet to be extended to the more expressive DLs that underpin Web ontology languages such as DAML+OIL and OWL DL [16], and it is not even clear if such an extension would be possible.

An alternative approach, and the one we investigate here, is to use a simple "direct" translation based on the standard first order semantics of DLs (see, e.g., [1]). Using this

[1] Briefly, the interaction between inverse roles, nominals and number restrictions causes problems for existing algorithms.

approach, an ontology/knowledge base (a set of DL axioms), is translated into a FO theory (a set of FO axioms). A DL reasoning task w.r.t. the knowledge base (KB) is then transformed into a FO task that uses the theory. Unlike methods such as Hustadt and Schmidt's functional translation, this does not result in a decision procedure for the DL. The direct translation approach can, however, be used to provide reasoning services (albeit without any guarantee of completeness) for the expressive DLs underlying Web ontology languages, DLs for which no effective decision procedure is currently known. Moreover, the translation approach can easily deal with language extensions such as SWRL (see Section 5).

In recent years, a number of highly efficient FO provers have been implemented [23, 33, 29]. These provers compete annually on a set of tasks, and the results are published [7]. One of the most successful general-purpose provers has been Vampire [29], and we have chosen this prover to use in our comparison.

2.1 Translation issues

We will only discuss the translation from DL to FOL as the correspondence between OWL DL and $\mathcal{SHOIN}\mathcal{D}_n^-$ is well known [17]. The translation ϕ maps DL concept C and role name R into unary and binary predicates $\phi_C(x)$ and $\phi_R(x, y)$ respectively. Complex concepts and axioms are mapped into FO formulae and axioms in the standard way [6, 1]. For example, subsumption and equivalence axioms are translated into, respectively, FO implication and equivalence (with the free variables universally quantified).

As an example, let's see a translation of a couple of concept and role axioms:

DL	FOL
$R \sqsubseteq S$	$\forall x \forall y(\phi_R(x, y) \rightarrow \phi_S(x, y))$
$C \equiv D \sqcap \exists R.(E \sqcup \forall S^-.F)$	$\forall x(\phi_C(x) \equiv \phi_D(x) \wedge \exists y(\phi_R(x, y) \wedge (\phi_E(y) \vee \forall x(\phi_S(x, y) \wedge \phi_F(x)))))$
$A \sqsubseteq \geqslant 3\, R.B$	$\forall x(\phi_A(x) \rightarrow \exists y_1 \exists y_2 \exists y_3(\phi_R(x, y_1) \wedge \phi_B(y_1) \wedge \phi_R(x, y_2) \wedge \phi_B(y_2) \wedge \phi_R(x, y_3) \wedge \phi_B(y_3) \wedge (y_1 \neq y_2) \wedge (y_2 \neq y_3) \wedge (y_1 \neq y_3))$
Transitive(T)	$\forall x \forall y \forall z(\phi_T(x, y) \wedge \phi_T(y, z) \rightarrow \phi_T(x, z))$

Simple DLs (like $\mathcal{ALC}$) can be translated into the FOL class $\mathcal{L}^2$ (the FOL fragment with no function symbols and only 2 variables), which is known to be decidable [22]. The above translations of the role inclusion axiom and concept equality axiom are, for example, in $\mathcal{L}^2$. When number restrictions are added to these DLs, they can be translated into $\mathcal{C}^2$—equivalent to $\mathcal{L}^2$ with additional "counting quantifiers"—which is also known to be decidable [12].

The FOL translation of more expressive description logics, e.g., with transitive roles ($\mathcal{SHIQ}$, OWL Lite and OWL DL) and/or complex role axioms ($\mathcal{RIQ}$ [19]), may lead to the introduction of three or more variables.[2] The above transitivity axiom for role T is an example of this case. FOL with three variables is known to be undecidable [6].

[2] In some cases, the effects of transitive roles can be axiomatised in $\mathcal{C}^2$ [35].

OWL DL also provides for XML schema *datatypes* [5], equivalent to a very simple form of *concrete domains* [18]. The minimum requirement for OWL DL reasoners is that they support `xsd:integer` and `xsd:string` datatypes, where support means providing a theory of (in)equality for integer and string values [27].

Our translation encodes the required datatype theory by mapping datatypes into predicates and data values into new constants. Lexically equivalent data values are mapped to the same constant, with integers first being canonicalised in the obvious way, and axioms are added that assert inequality between all the string and integer data constants introduced. If a data value `DV` and a datatype `DT` are mapped to DV and DT respectively, and `DV` is of type `DT`, then an axiom $DT(DV)$ is also added. As the `xsd:integer` and `xsd:string` interpretation domains are disjoint, we add an axiom to that effect. Finally, we add an axiom asserting the disjointness of the datatype domain (the set of data values) and the abstract domain (the set of individuals).

In accordance with the OWL DL semantics, other "unsupported" data types are treated opaquely, i.e., data values are mapped to the same constant if they are lexically identical, but no other assumptions are made (we do *not* assume inequality if the lexical forms are not identical) [27].

2.2 Key reasoning tasks.

Given a DL *Knowledge Base* $\mathcal{K}$ (equivalently an OWL DL *ontology*) consisting of a set of axioms about concepts, roles, individuals and (possibly) datatypes, key reasoning tasks w.r.t. $\mathcal{K}$ include: Knowledge Base (KB) consistency (do the axioms in $\mathcal{K}$ lead to a contradiction); concept consistency (is it consistent for a concept to have at least one instance); concept subsumption (are the instances of one concept necessarily instances of another); instance checking (does a given class contain a given instance); instance retrieval (find all KB instances for a given class); classification (compute the subsumption partial ordering of concept names in $\mathcal{K}$); and entailment (does $\mathcal{K}$ entail all of the axioms in another KB). All of these tasks are reducible to KB consistency or, equivalently, concept consistency w.r.t. a KB [17] (in fact they are reducible to concept consistency w.r.t. an empty KB via internalisation [1]).

The tests presented here include several classification tasks, and a set of OWL DL consistency and entailment tasks. Each classification task transforms into a (usually quite large) set of concept subsumption/consistency tasks.

As a standard DL reasoner we will use FaCT++ version 0.85 beta. This system is a next generation of the well-known FaCT reasoner [15], being developed as part of the EU WonderWeb project (see `http://wonderweb.semanticweb.org/`). This implementation is based on the same tableaux algorithms as the original FaCT, but has a different architecture and is written in C++ instead of LISP. The goal of developing FaCT++ was to create a modern reasoner for complex DLs (like $\mathcal{SHIQ}$ and OWL DL) with good performance, improved extensibility and internal data structures that are better able to handle very large knowledge bases.

As an FOL prover we used Vampire v6 [29]. Vampire is a general-purpose FOL prover developed by Andrei Voronkov and Alexandre Riazanov. Given a set of first-order formulas in the full FOL syntax, Vampire transforms it into an equisatisfiable set

of clauses, and then tries to demonstrate inconsistency of the clause set by saturating it with ordered resolution and superposition (see [3, 24]). If the saturation process terminates without finding a refutation of the input clause set, it indicates that the clause set, and therefore the original formula set, is satisfiable, provided that the variant of the calculus used is refutationally complete and that a fair strategy[3] has been used for saturation. The main input format of Vampire is the TPTP syntax [31], although a parser for a subset of KIF [11] has been added recently.

Vampire is one of the most competitive general-purpose provers, which is demonstrated by its performance in the last five CASC competitions [7]. However, we would like to stress that Vampire uses no optimisations targeting specific needs of ontological reasoning. Moreover, most of the strong features of the implementation are not very useful in the experiments presented here. In particular, Vampire's implementation is highly optimised towards finding refutations, often at the cost of using incomplete procedures, whereas the majority of subsumption tests in our current experiment require the detection of satisfiability. Also, the needs of light-weight reasoning, when one needs to solve relatively easy problems very fast, have never been addressed. Most of the core implementation features, such as a number of complex term indexing techniques, introduce noticeable overhead during the first seconds of a run, which only pays off on reasonably hard problems.

3 Ontology reasoning comparison

3.1 Test ontologies

We used three ontologies (KBs) in the test: the Tambis ontology[4] has a small size and very simple structure; the Wine ontology[5] has a similar size, but a much more complex structure and 150 general concept inclusion axioms (GCIs)[6] [1]; the Galen ontology[7] has a very large size with quite a simple concept language (only conjunction and existential restrictions), but with transitive roles and more than 400 GCIs. Table 1 gives the size of the ontologies along with the number of (positive and negative) subsumption tests performed by FaCT++ during classification.

Table 1. Ontologies used for the comparison

Ontology	Concepts	Roles	Total subs	Pos subs	Neg subs
Tambis	345	107	597	25	572
Wine	346	16	2221	465	1756
Galen	2749	207	25631	3561	22070

[3] i.e., all generated clauses are eventually processed

[4] A biochemistry ontology developed in the Tambis project [4].

[5] A wine and food ontology which forms part of the OWL test suite [25].

[6] A GCI is an axiom of the form $C \sqsubseteq D$, where C may be an arbitrarily complex class description. The presence of GCIs can significantly decreases the performance of a tableaux reasoner.

[7] A medical terminology ontology developed in the Galen project [28].

3.2 Experimental methodology

We used the following comparison scheme. For a given ontology, the DL reasoner performed the classification process. In the case where a subsumption test is necessary, a task was also generated for the FO prover, and the result of the DL testing was noted for future comparison. After finishing the classification process, the set of problems that were solved had been generated. The FO prover was then run on this set of problems. The results were recorded and then compared with the DL ones.

In the experiments with Galen, Wine and Tambis we used the default settings of Vampire with the following two adjustments. Firstly, the default saturation procedure may apply an incomplete strategy, so we replaced it by a complete one (`--main_alg otter`). Secondly, the literal selection function, which is one of the most important resolution strategy parameters, was set to simulate positive hyperresolution (`--selection 2`), as this strategy showed the best results on a training set of tasks randomly selected from the given problems. All tests with Vampire and FaCT++ were made on Pentium III, 1000 Mhz running Linux. We ran Vampire with a time limit of 300 seconds and a memory limit of 350Mb per task.

Note that the actual number of comparisons performed in order to classify an ontology depends on the DL system's classification algorithm. If the algorithm is optimised (i.e., exploits concept structure, results of previous operations, etc.), then the number of (logical) subsumption tests may be much smaller than the number required by a naive (unoptimised) algorithm. In the FaCT++ classifier, for example, the number of actually performed subsumption tests is typically only 0.5-2% of those that would be required in a "brute force" implementation (see [2]). Moreover, relatively few of these tests give a positive result (i.e., one in which a subsumption is proved), because most subsumptions are "obvious", and do not need to be computed using a logical subsumption test.

The comparisons were generated by FaCT++ running in its standard (optimised) mode. This resulted in all of the general axioms in the test ontologies being *absorbed* [14], i.e., axioms of the form $C \sqsubseteq D$ (where C is a complex concept) were re-written as axioms of the form $CN \sqsubseteq D'$ (where CN is a primitive concept name). Absorbed axioms can be more efficiently dealt with by FaCT++'s tableaux algorithm.

3.3 Basic translation tests

Results using the basic translation (see Section 2.1 for details) are presented in Table 2. The number of (solved) tasks is given, along with the average time (in seconds) per solved task. The results are broken down into positive tasks (ones where subsumption is proved) and negative tasks (ones where non-subsumption is proved). Tasks that are not solved don't influence this parameter.

Vampire can solve every positive task for Tambis, 93% for Wine and 99% for Galen. It could not solve any negative subsumption task for Wine, but it solved 78% of negative tasks for Galen.

3.4 Relevant-only translation tests

One problem with the results presented in the previous section is that Vampire receives *all* of the axioms that occur in the ontology when usually only a small fraction of them

Table 2. Results for basic translation

Ontology	Task type	Tasks	Solved tasks		Av. time (secs)	
			FaCT++	Vampire	FaCT++	Vampire
Tambis	pos	25	25	25	< 0.01	0.31
	neg	572	572	572	< 0.01	0.31
Wine	pos	465	465	433	< 0.01	41.94
	neg	1756	1756	0	0.037	—
Galen	pos	3561	3561	3536	< 0.01	23.75
	neg	22070	22070	17260	0.027	24.20

are actually relevant to a given subsumption problem. Unlike FaCT++, Vampire is not optimised to deal efficiently with large numbers of irrelevant axioms, and so is at a significant disadvantage.

An obvious way to correct this situation is to remove all irrelevant information from the FO task given to Vampire. We call an axiom *irrelevant* to a subsumption test $C \sqsubseteq D$ if we can easily show (i.e., via a syntactic analysis) that removing it from the ontology would not affect the interpretation of either C or D; other axioms are called *relevant*. Note that not every "relevant axiom" really *will* affect the computation of the test $C \sqsubseteq D$, but we cannot (easily) rule out the possibility that it *may* affect the computation. In the rest of paper, we will say that an FO-translation is *relevant-only* if it contains only FO-translations of axioms relevant (in the above sense) to the given subsumption test.

A concept or role expression *depends* on every concept or role that occurs in it, and a concept or role C depends on a concept or role D if D occurs in the definition of C. In addition, a concept C depends on every GCI in the ontology. *Relevance* is the transitive closure of depends. An algorithm for computing relevant information is quite straightforward and is described in detail in [36].

Table 3. Results for relevant-only translation

Ontology	Task type	Tasks	Solved tasks		Av. time (secs)	
			FaCT++	Vampire	FaCT++	Vampire
Tambis	pos	25	25	25	< 0.01	0.03
	neg	572	572	572	< 0.01	0.03
Wine	pos	465	465	461	< 0.01	3.44
	neg	1756	1756	1243	0.037	14.32
Galen	pos	3561	3561	3561	< 0.01	2.48
	neg	22070	22070	17271	0.027	1.73

Results using the relevant-only translation are presented in Table 3. Note that here only Vampire running times are changed (the additional overhead for creating relevant-only tasks is negligible, and we did not include it in the comparison).

The removal of irrelevant information results in a big improvement for Vampire. For the large but quite simple Galen ontology, the number of solved tests increased, and the average time was significantly reduced. In particular, Vampire solved all positive

tests. For the complex Wine ontology, the majority of negative subsumption tests were solved, whereas none were solved before.

3.5 Non-absorbed translation tests

In our test ontologies, all GCIs were successfully absorbed. In general, however, not every GCI can be absorbed. It is well known that non-absorbed GCIs lead to a significant degradation in the performance of DL reasoners [15], so we decided to compare the effect of non-absorbed GCI's on Vampire and FaCT++ by switching off the absorption optimisation in FaCT++. The results of this test are presented in Table 4.

Table 4. Results for non-absorbed translation

Ontology	Task type	Tasks	Solved tasks		Av. time (secs)	
			FaCT++	Vampire	FaCT++	Vampire
Tambis	pos	25	25	25	< 0.01	0.30
	neg	572	572	572	< 0.01	0.31
Wine	pos	465	465	465	0.035	4.85
	neg	1756	1756	0	1.33	—
Galen	pos	3561	0	3550	—	18.38
	neg	22070	0	17263	—	17.95

In this test Vampire can solve all positive tasks with the smaller ontologies and 99% with the Galen one. It could not solve any negative subsumption tasks for Wine, but 78% of Galen's negative tasks were solved. FaCT++ could not solve any subsumption tests for the Galen ontology, and shows significantly (up to 2 orders) worse results for the Wine ontology. This test also demonstrates that a FO prover may be able to solve some subsumption tests that are very hard for a tableaux based DL reasoner.

It is interesting to note that the results for Vampire are similar to those for the basic translation given in Table 2, with the average solution time for the positive Wine problems actually being 9 times longer for the basic translation. This seems to be due to the increased number of complex clauses generated by the absorption optimisation (as we can see in Table 6, the fraction of non-horn clauses grows 4 times in the absorbed case for the Wine ontology), which makes the proving process harder for Vampire.

An example illustrating how absorption can increase the number of non-horn clauses is given in Table 5. Here C, D, E and F are concept names. The GCI $C \sqsubseteq F$ is absorbed into the axiom $D \sqsubseteq F \sqcup E$, and the corresponding formula becomes non-horn.

3.6 Translation result profiles

To give the reader an idea of the syntactic complexity of the first-order proof tasks which Vampire has to deal with in our experiments, we present in Table 6 the integral syntactic characteristics of the clause sets obtained by clausifying the input provided by FaCT++. In addition to some standard parameters (i.e. number of non-horn clauses and number of clauses with positive equalities), we count the number of so-called *chain*

Table 5. Explanation of non-horn clause introduction

Ontology	DL axiom	FO axiom	Clause type
Non absorbed	$C \equiv D \sqcap \neg E$	$\forall x(\phi_C(x) \equiv (\phi_D(x) \wedge \neg\phi_E(x)))$	
	$C \sqsubseteq F$	$\forall x(\phi_C(x) \rightarrow \phi_F(x))$	horn
Absorbed	$C \equiv D \sqcap \neg E$	$\forall x(\phi_C(x) \equiv (\phi_D(x) \wedge \neg\phi_E(x)))$	
	$D \sqsubseteq F \sqcup E$	$\forall x(\phi_D(x) \rightarrow (\phi_F(x) \vee \phi_E(x)))$	non-horn

clauses, i.e., clauses of the form $\pm p(x_1, \ldots, x_n) \vee \pm q(x_1, \ldots, x_n)$, where p and q are different predicates. This is not directly related to our current experiments, but the high proportion of chain clauses in our tests supports the thesis of applicability of *chain resolution* [34] to terminological reasoning, and may be a valuable observation for future work.

Table 6. Syntactic profiles of the test suites

	Tambis			Galen			Wine		
translation type	Bas.	Rel.	N.A.	Bas.	Rel.	N.A.	Bas.	Rel.	N.A.
num. of clauses	576	42	576	6111	1715	6146	3354	1223	3123
% nonhorn	0	0	0	0	0	0	12	32	3
% chain	24	29	24	60	72	61	64	71	61
% with pos. eq.	5	4	5	2	3	2	7	8	8
lit. per clause	2.05	2.03	2.05	2.71	3.88	2.59	2.58	3.65	2.12
symbols per lit.	2.4	2.35	2.4	2.52	2.57	2.51	2.54	2.68	2.42

4 OWL tests comparison

The W3C WebOnt working group[8] have been responsible for the definition of the OWL Web Ontology Language. A key facet of the WebOnt activity has been to define a suite of OWL Tests [25]. These consist of a number of tasks categorised according to their type (consistency, entailment, etc.) and the expected result (positive or negative). In the main, the problems are trivial in terms of reasoning, and are intended primarily to illustrate the features of the language. A number of tests have, however, been translated from other test collections (notable the DL98 tests [9]), and these provide more challenging problems for reasoners, testing both correctness (e.g., termination on problems with no finite interpretation) and efficiency (e.g., of searching large spaces of possible interpretations).

Because no existing DL reasoner can deal with the complete OWL language, we decided to use Vampire to try to prove (or disprove) the correctness of (some of) the OWL tests. Not all of the OWL tests were attempted as some are, e.g., designed to test parsers or OWL Full reasoners, and some use unsupported datatypes. The tests

[8] `http://www.w3.org/2001/sw/WebOnt`

we attempted here are the consistency tests (both positive and negative) for OWL DL ontologies that use no datatypes other than `xsd:string` or `xsd:integer`.

In order to use Vampire with these tests, the OWL DL ontologies were translated into FO axioms as described in Section 2.1; we then tried to prove the inconsistency (consistency) of the resulting FO theory. In principle it would also be possible to use Vampire with the positive entailment tests by trying to prove the inconsistency of the premise theory augmented with the negated conclusion theory, but this would require a more sophisticated translation in order to deal correctly with anonymous individuals in the negated conclusion theory [17].

The current conversion strategy ignores OWL annotations, i.e., comments that are intended to have no formal semantics. This has no impact on the Consistency and Inconsistency tests. For entailment tests, however, further investigation would be needed as annotations do have an impact on entailment (in particular negative entailment) [27].

4.1 Test Results

We categorise the test results as follows. *Pass*—the prover returned a definitive result which is as expected. *Fail*—the prover returned a definitive result which is not as expected. *Unknown*—the prover failed as a result of technical problems with the translation or with Vampire. *Timeout*—the prover ran out of time (the prover is currently allowed up to 300 seconds to find a proof).

Strictly speaking, any result other than pass should be taken as a failure. A wrong answer is, however, clearly a much more serious failure than an inability to find a proof (timeout) or an inability to deal with the problem (unknown).

Table 7. OWL test results

Type	Attempted	Pass	Fail	Unknown	Timeout	Pass (%)
Inconsistent	66	64	0	1	1	97%
Consistent	46	43	0	1	2	93%
Total	112	107	0	2	3	96%

A summary of the current results is shown in Table 7. As can be seen, Vampire was able to pass over 95% of the tests. Moreover, of the 2 tests returning Unknown, one was due to the presence of large cardinality constraints (with values $>$500) in the original problem. The number of variables introduced in the translation results in a very large source file for the problem, causing technical difficulties in parsing and converting the test.

5 Extension to SWRL

An attractive feature of the use of a first order prover is its extensibility. The proposed "Semantic Web Stack" includes further layers that build on the Ontology or Logical

layer provided by OWL (or similar languages). These layers are likely to add expressivity that moves us out of the fragments of FOL supported by Description Logic reasoners, so alternative techniques will be required to handle these extensions.

One such proposed extension to OWL is the Semantic Web Rule Language (SWRL) [13], which extends OWL with Horn-like rules. Note that SWRL is different from languages such as CARIN [21], where the combination of a relatively inexpressive DL ($\mathcal{ALCNR}$), and rather severe syntactic restrictions on rules, are used to ensure the decidability of the combined language. In contrast, SWRL's underlying logic is much more expressive, and the form of rules is much less restrictive, with the result that the combined language is no longer decidable. Using the translation approach, however, we can easily extend our first-order translation to incorporate these rules and provide a simple implementation of a SWRL reasoner.

5.1 Translating SWRL Rules

Rules in SWRL are of the form:

$$B_1, \ldots, B_m \rightarrow H_1, \ldots, H_n$$

where each of the B_i or H_j are rule *atoms*. Possible rule atoms are shown in Table 8. Here C is an OWL class description, R an OWL property and i and j are either OWL individual names or SWRL variables. We consider a simplification of the proposal in

Table 8. Rule Atoms

Atom	**Type**
`C(i)`	Class Atom
`R(i,j)`	Property Atom
`i==j`	Equality Atom
`i!=j`	Inequality Atom

[13], where C must be a class name (rather than arbitrary class descriptions), and R must be an object property. The first of these restrictions does not affect the expressivity of the language, as new class names can be introduced into the ontology to represent any complex descriptions required in rules. The restriction to object properties simplifies our implementation, but the translation we describe could easily be extended to handle data valued properties.

The translation of rules exactly follows the semantics of the rules as given in [13]. Each rule is translated as an implication, and any free variables in the rule are assumed to be universally quantified. Thus a rule:

$$B_1, \ldots, B_m \rightarrow H_1, \ldots, H_n$$

is translated to an axiom:

$$\forall x_1, x_2, \ldots, x_k . T(B_1) \wedge \ldots \wedge T(B_m) \rightarrow T(H_1) \wedge \ldots \wedge T(H_n)$$

Table 9. Rule Atom Translation

Atom	Translation
`C(i)`	$C(i)$
`R(i,j)`	$R(i,j)$
`i==j`	i=j
`i!=j`	$i \neq j$

where $x_1, x_2, \ldots, x_k$ are all the variables occurring in the B_i and H_j.

Translation of atoms is trivial and is shown in Table 9. Combining this translation with the translation from OWL to FOL described in Section 2 provides us with a prototype implementation of a SWRL reasoner. Given an ontology and a collection of rules relating to that ontology, we translate the ontology to FOL, and then add the FOL axioms generated by translating the rules. The resulting theory is passed to a FO prover, where it can be used for reasoning tasks such as satisfiability checking and instance checking. Of course the effectiveness of such a naive approach is open to question, but this does at least provide us with a reasoner *for the complete language* (rather than for some fragment that can be handled by a rule or DL reasoner) that can be used for illustrative and test purposes.

5.2 Examples

The proposed rule language extends the expressivity of OWL, in particular allowing us to provide extra information about properties. A standard example of this is the following definition of "uncle", which cannot be expressed in OWL alone:

$$\texttt{hasParent(?x,?y), hasSibling(?y, ?z), Male(?z)} \Rightarrow \texttt{hasUncle(?x,?z)}$$

If our ontology additionally includes the axiom and facts (expressed here using standard DL syntax):

$$\texttt{Uncle} \equiv \exists \texttt{hasUncle}^{-}.\top$$
$$\langle \texttt{Robert}, \texttt{Paul} \rangle : \texttt{hasParent}$$
$$\langle \texttt{Paul}, \texttt{Ian} \rangle : \texttt{hasSibling}$$

then the reasoner can infer not only $\texttt{hasUncle}(\texttt{Robert}, \texttt{Ian})$, but also that `Ian` is an instance of the `Uncle` class.

Another interesting aspect of the language is illustrated by the following rule:

$$\texttt{Beer(?x)} \Rightarrow \texttt{Happy(Sean)}$$

This expresses the fact that for any instances of the class `Beer`, `Sean` must be an instance of `Happy`. This effectively allows us to express an existential quantification over the class `Beer`: if we can prove the existence of an instance of this class, then `Sean` will be `Happy`. Note that we do not actually have to provide a name for such an instance. For example, if our ontology includes the fact:

$\texttt{Sean} : \exists \texttt{drinks}.\texttt{Beer}$

then the reasoner can infer that Sean must be Happy as we now know that there exists *some* instance of Beer—even though this instance is unnamed.

6 Discussion

As can be seen from the results, the performance of Vampire is much worse than that of FaCT++ when tested with reasoning tasks derived from a naive translation of subsumption tests (w.r.t. an ontology). When a suitably optimised translation is used, however, the performance of Vampire improves dramatically: for the small ontology it is comparable with that of FaCT++, although for the more complex Galen and Wine ontologies it is still in the order of 100 times slower (and this does not take into consideration the tests which Vampire is unable to solve within the 300s time limit).

Vampire is able to solve the vast majority of tasks within the time limit, but for the complex ontologies it still fails on a significant number of negative tests (non-subsumption). Unfortunately, the vast majority of tests performed during ontology classification are negative tests. It should also be pointed out that performing each subsumption test in isolation puts Vampire at a considerable disadvantage, as fixed startup costs are incurred in every test, and information from previous tests cannot be reused.

The performance of Vampire is sufficiently encouraging to suggest that further investigations of FO theorem proving techniques for OWL ontology derived tasks would be worthwhile. The FO tasks generated in the tests are in the TPTP format [32], which is a de-facto standard for the theorem proving community, making it easy to use other FO provers in a similar comparison. Given that the performance of FO provers can vary greatly depending on the type of problem, it may be that another FO prover would give better performance on DL subsumption reasoning tasks. On the other hand, the performance gap with respect to FaCT++ is sufficiently large that the designers of FO provers might be encouraged to consider adding some DL reasoning oriented optimisations to their systems.

Although the results presented here do not suggest that FO provers might be used to replace dedicated DL reasoners, they do illustrate that a FO prover can be a useful tool for ontology reasoning. A FO prover might, for example, be used as part of the infrastructure for developing new algorithms and reasoners for various languages, in particular for debugging, prototyping, and for checking the status of newly developed test problems (such as those in the OWL test suite).

It might also be useful to use a FO prover in a hybrid tool for dealing with very expressive languages, e.g., for OWL DL, where reasoning with the complete language is beyond the scope of existing DL algorithms, or for SWRL, where the complete language no longer corresponds to a DL (or to any other decidable fragment of FOL). In these cases, it would be possible to use a FO prover to compute some or all of the relevant inferences. Although there would inevitably be problems with the speed of response, and with incompleteness, there would still be an improvement in performance given that existing DL reasoners currently can't deal with these situations *at all*.

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002.
2. Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An empirical analysis of optimization techniques for terminological representation systems. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 270–281. Morgan Kaufmann, Los Altos, 1992.
3. L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.
4. P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. Tambis: Transparent access to multiple bioinformatics information sources: an overview. In *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology, ISMB98*, pages 25–34. AAAI Press, 1998.
5. Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes. W3C Recommendation, May 2001. `http://www.w3.org/TR/xmlschema-2/`.
6. Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.
7. The CASC web page. `http://www.cs.miami.edu/~tptp/CASC/`.
8. Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. W3C Recommendation, 10 February 2004. `http://www.w3.org/TR/owl-ref/`.
9. DL'98 test suite. `http://dl.kr.org/dl98/comparison/`.
10. Harald Ganzinger and Hans de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 295–304, 1999.
11. M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format Version 3.0 Reference Manual. Technical Report Logic Group Report Logic-92-1, Stanford University, 2001. `http://logic.stanford.edu/kif/Hypertext/kif-manual.html`.
12. E. Gradel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proceedings of LICS 1997*, pages 306–317, 1997.
13. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. `http://www.daml.org/2003/11/swrl/`, 2003.
14. I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 285–296, 2000.
15. Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.
16. Ian Horrocks and Peter F. Patel-Schneider. The generation of DAML+OIL. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 30–35. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-49/`, 2001.
17. Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 17–29. Springer, 2003.

18. Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204, 2001.
19. Ian Horrocks and Ulrike Sattler. The effect of adding complex role inclusion axioms in description logics. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 343–348. Morgan Kaufmann, Los Altos, 2003.
20. U. Hustadt and R. A. Schmidt. MSPASS: Modal reasoning by translation and first-order resolution. In R. Dyckhoff, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference (TABLEAUX 2000)*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 67–71. Springer, 2000.
21. Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *European Conference on Artificial Intelligence*, pages 323–327, 1996.
22. M. Mortimer. On languages with two variables. *Zeitschr. für math. Logik und Grundlagen der Math.*, 21:135–140, 1975.
23. Max Moser, Ortrun Ibens, Reinhold Letz, Joachim Steinbach, Christoph Goller, Johann Schumann, and Klaus Mayr. SETHEO and e-SETHEO - the CADE-13 systems. *Journal of Automated Reasoning*, 18(2):237–246, 1997.
24. R. Nieuwenhuis and A. Rubio. Paramodulation-Based Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 7, pages 371–443. Elsevier Science, 2001.
25. OWL test suite. `http://www.w3.org/TR/owl-test/`.
26. M. Paramasivam and David A. Plaisted. Automated deduction techniques for classification in description logic systems. *J. of Automated Reasoning*, 20(3):337–364, 1998.
27. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/owl-semantics/`.
28. A. L. Rector, W. A. Nowlan, and A. Glowinski. Goals for concept representation in the GALEN project. In *Proc. of the 17th Annual Symposium on Computer Applications in Medical Care (SCAMC'93)*, pages 414–418, Washington DC, USA, 1993.
29. A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.
30. Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.
31. G. Sutcliffe and C. Suttner. The TPTP Problem Library. TPTP v. 2.4.1. Technical report, University of Miami, 2001.
32. G. Sutcliffe and C. Suttner. The TPTP Problem Library. TPTP v. 2.4.1. Technical report, University of Miami, 2001.
33. T. Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199–204, 1997.
34. Tanel Tammet. Extending Classical Theorem Proving for the Semantic Web. In R. Volz, S. Decker, and I. Cruz, editors, *Proceedings of the First International Workshop on Practical and Scalable Semantic Systems*, October 2003. `http://km.aifb.uni-karlsruhe.de/ws/psss03/proceedings`.
35. Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.
36. Dmitry Tsarkov and Ian Horrocks. DL reasoner vs. first-order prover. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, volume 81 of *CEUR (`http://ceur-ws.org/`)*, pages 152–159, 2003.