OWL Rules: A Proposal and Prototype Implementation

Ian Horrocks Peter F. Patel-Schneider Sean Bechhofer
Dmitry Tsarkov

February 28, 2005

Abstract

Although the OWL Web Ontology Language adds considerable expressive power to the Semantic Web it does have expressive limitations, particularly with respect to what can be said about properties. We present SWRL (the Semantic Web Rules Language), a Horn clause rules extension to OWL that overcomes many of these limitations. SWRL extends OWL in a syntactically and semantically coherent manner: the basic syntax for SWRL rules is an extension of the abstract syntax for OWL DL and OWL Lite; SWRL rules are given formal meaning via an extension of the OWL DL model-theoretic semantics; SWRL rules are given an XML syntax based on the OWL XML presentation syntax; and a mapping from SWRL rules to RDF graphs is given based on the OWL RDF/XML exchange syntax. We discuss the expressive power of SWRL, showing that the ontology consistency problem is undecidable, provide several examples of SWRL usage, and discuss a prototype implementation of reasoning support for SWRL.

1 Introduction

The OWL Web Ontology Language [47] adds considerable expressive power to the Semantic Web. However, for a variety of reasons (see `http://lists.w3.org/Archives/Public/www-webont-wg/` and [20]), including retaining the decidability of key inference problems in OWL DL and OWL Lite, OWL has expressive limitations. These restrictions can be onerous in some application domains, for example in describing web services, where it may be necessary to relate inputs and outputs of composite processes to the inputs and outputs of their component processes [51], or in medical informatics, where it may be necessary to transfer characteristics across partitive properties [39].

Many of the limitations of OWL stem from the fact that, while the language includes a relatively rich set of class constructors, the language provided for talking about properties is much weaker. In particular, there is no composition constructor, so it is impossible to capture relationships between a composite property and another (possibly composite) property. The standard example here is the obvious relationship between the composition of the "parent" and "brother" properties and the "uncle" property.

One way to address this problem would be to extend OWL with a more powerful language for describing properties. For example, a decidable extension of the description logics underlying OWL DL to include the use of composition in subproperty axioms has already been investigated [22, 23]. In order to maintain decidability, however,

the usage of the constructor is limited to axioms of the form $P \circ Q \sqsubseteq P$, i.e., axioms asserting that the composition of two properties is a subproperty of one of the composed properties. This means that complex relationships between composed properties cannot be captured—in fact even the relatively simple "uncle" example cannot not be captured (because "uncle" is not one of "parent" or "brother").

An alternative way to overcome some of the expressive restrictions of OWL would be to extend it with some form of "rules language". In fact adding rules to description logic based knowledge representation languages is far from being a new idea. Several early description logic systems, e.g., Classic [38, 8], included a rule language component. In these systems, however, rules were given a weaker semantic treatment than axioms asserting sub- and super-class relationships; they were only applied to individuals, and did not affect class based inferences such as the computation of the class hierarchy. More recently, the CARIN system integrated rules with a description logic in such a way that sound and complete reasoning was still possible [28]. This could only be achieved, however, by using a rather weak description logic (*much* weaker than OWL), and by placing severe syntactic restrictions on the occurrence of description logic terms in the (heads of) rules. Similarly, the DLP language proposed in [14] is based on the intersection of a description logic with horn clause rules; the result is obviously a decidable language, but one that is necessarily less expressive than either the description logic or rules language from which it is formed.

In this paper we show how a simple form of Horn-style rules can be added to the OWL language in a syntactically and semantically coherent manner, the basic idea being to add such rules as a new kind of axiom in OWL DL. We show (in Section 3) how the OWL abstract syntax in the OWL Semantics and Abstract Syntax document [37] can be extended to provide a formal syntax for these rules, and (in Section 4) how the direct OWL model-theoretic semantics for OWL DL can be extended to provide a formal meaning for OWL ontologies including rules written in this abstract syntax. We will also show (in Section 5) how OWL's XML presentation syntax can be modified to deal with the proposed rules.

The extended language was originally called ORL (the OWL Rules Language), but is now much better known as SWRL (the Semantic Web Rules Language), a name that was coined when the Joint US/EU ad hoc Agent Markup Language Committee[1] developed a W3C members submission based on ORL.[2] Although SWRL includes some additional features (mainly related to datatypes and predicates) and has some minor syntactic differences, we will refer to the language described here as SWRL.

SWRL is considerably more powerful than either OWL DL or Horn rules alone. We will show (in Section 6) that the key inference problems (e.g., ontology consistency) for SWRL are undecidable, and (in Section 7) provide examples that utilise the power of the combined languages.

In Section 8 we show how OWL's RDF syntax can be extended to deal with rules, and in Sections 9 and 10 we discuss how reasoning support for SWRL might be provided. Finally (in Section 11), we summarise the main features of the SWRL proposal and suggest some directions for future work.

[1] `http://www.daml.org/committee/`

[2] `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`

2 Overview

The basic idea of the proposal is to extend OWL DL with a form of rules while maintaining maximum backwards compatibility with OWL's existing syntax and semantics. To this end, we add a new kind of axiom to OWL DL, namely Horn clause rules, extending the OWL abstract syntax and the direct model-theoretic semantics for OWL DL [37] to provide a formal semantics and syntax for OWL ontologies including such rules.

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The informal meaning of a rule can be read as: whenever (and however) the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

Both the antecedent (body) and consequent (head) of a rule consist of zero or more atoms. Atoms can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL DL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values. Atoms are satisfied in extended interpretations (to take care of variables) in the usual model-theoretic way, i.e., the extended interpretation maps the variables to domain elements in a way that satisfies the description, property, sameAs, or differentFrom, just as in the regular OWL model theory.

Multiple atoms in an antecedent are treated as a conjunction. An empty antecedent is thus treated as trivially true (i.e. satisfied by every interpretation), so the consequent must also be satisfied by every interpretation.

Multiple atoms in a consequent are treated as separate consequences, i.e., they must all be satisfied. In keeping with the usual treatment in rules, an empty consequent is treated as trivially false (i.e., not satisfied by any extended interpretation). Such rules are satisfied if and only if the antecedent is not satisfied by any extended interpretation. Note that rules with multiple atoms in the consequent could easily be rewritten (by applying standard rules of distributivity) into multiple rules each with an atomic consequent.

It is easy to see that OWL DL becomes undecidable when extended in this way as rules can be used to simulate role value maps [46] and make it easy to encode known undecidable problems as a SWRL ontology consistency problem (see Section 6).

3 Abstract Syntax

The syntax for SWRL in this section abstracts from any exchange syntax for OWL and thus facilitates access to and evaluation of the language. This syntax extends the abstract syntax of OWL described in the OWL Semantics and Abstract Syntax document [37].

Like the OWL abstract syntax, we will specify the abstract syntax for rules by means of a version of Extended BNF, very similar to the Extended BNF notation used for XML [52]. In this notation, terminals are quoted; non-terminals are not quoted. Alternatives are either separated by vertical bars (|) or are given in different productions. Components that can occur at most once are enclosed in square brackets ([...]); components that can occur any number of times (including zero) are enclosed in braces ({...}). Whitespace is ignored in the productions given here.

Names in the abstract syntax are RDF URI references [27]. These names may be abbreviated into qualified names, using one of the following namespace names:

rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#
owl	http://www.w3.org/2002/07/owl#

The meaning of each construct in the abstract syntax for rules is informally described when it is introduced. The formal meaning of these constructs is given in Section 4 via an extension of the OWL DL model-theoretic semantics [37].

3.1 Rules

From the OWL Semantics and Abstract Syntax document [37], an OWL ontology in the abstract syntax contains a sequence of annotations, axioms, and facts. Axioms may be of various kinds, for example, subClass axioms and equivalentClass axioms. This proposal extends axioms to also allow rule axioms, by adding the production:

axiom ::= rule

Thus a SWRL ontology could contain a mixture of rules and other OWL DL constructs, including ontology annotations, axioms about classes and properties, and facts about OWL individuals, as well as the rules themselves.

A rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a (possibly empty) set of atoms. Just as for class and property axioms, rule axioms can also have annotations. These annotations can be used for several purposes, including giving a label to the rule by using the rdfs:label annotation property.

rule ::= 'Implies('{annotation} antecedent consequent')'
antecedent ::= 'Antecedent('{atom}')'
consequent ::= 'Consequent('{atom}')'

Informally, a rule may be read as meaning that if the antecedent holds (is "true"), then the consequent must also hold. An empty antecedent is treated as trivially holding (true), and an empty consequent is treated as trivially not holding (false). Non-empty antecedents and consequents hold iff all of their constituent atoms hold. As mentioned above, rules with multiple consequents could easily be rewritten (using standard rules of distributivity) into multiple rules each with a single atomic consequent.

Atoms in rules can be of the form C(x), P(x,y), Q(x,z), sameAs(x,y) or differentFrom(x,y), where C is an OWL DL description, P is an OWL DL *individual-valued* Property, Q is an OWL DL *data-valued* Property, x,y are either variables or OWL individuals, and z is either a variable or an OWL data value. In the context of OWL Lite, descriptions in atoms of the form C(x) may be restricted to class names.

atom ::= description '(' i-object ')'
| individualvaluedPropertyID '(' i-object i-object ')'
| datavaluedPropertyID '(' i-object d-object ')'
| sameAs '(' i-object i-object ')'
| differentFrom '(' i-object i-object ')'

Informally, an atom C(x) holds if x is an instance of the class description C, an atom P(x,y) (resp. Q(x,z)) holds if x is related to y (z) by property P (Q), an atom sameAs(x,y) holds if x is interpreted as the same object as y, and an atom differentFrom(x,y) holds if x and y are interpreted as different objects.

Atoms may refer to individuals, data literals, individual variables or data variables. Variables are treated as universally quantified, with their scope limited to a given rule. As usual, only variables that occur in the antecedent of a rule may occur in the consequent (a condition usually referred to as "safety").

i-object ::= i-variable | individualID
d-object ::= d-variable | dataLiteral

i-variable ::= 'I-variable(' URIreference ')'
d-variable ::= 'D-variable(' URIreference ')'

3.2 Human Readable Syntax

While the abstract Extended BNF syntax is consistent with the OWL specification, and is useful for defining XML and RDF serialisations, it is rather verbose and not particularly easy to read. In the following we will, therefore, often use a relatively informal "human readable" form similar to that used in many published works on rules.

In this syntax, a rule has the form:

$$\mathsf{antecedent} \rightarrow \mathsf{consequent},$$

where both antecedent and consequent are conjunctions of atoms written $a_1 \wedge \ldots \wedge a_n$. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., $?x$). Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

$$\mathsf{parent}(?a, ?b) \wedge \mathsf{brother}(?b, ?c) \rightarrow \mathsf{uncle}(?a, ?c). \quad (1)$$

If John has Mary as a parent and Mary has Bill has a brother, then this rule requires that John has Bill as an uncle. Using the abstract syntax described in Section 3.1, this rule would have been written as:

```
Implies(Antecedent(parent(I-variable(a) I-variable(b))
                   brother(I-variable(b) I-variable(c)))
        Consequent(uncle(I-variable(a) I-variable(c)))).
```

4 Direct Model-Theoretic Semantics

The model-theoretic semantics for SWRL is a straightforward extension of the semantics for OWL DL given in [37]. The basic idea is that we define *bindings*—extensions of OWL interpretations that also map variables to elements of the domain in the usual manner. A rule is satisfied by an interpretation iff every binding that satisfies the antecedent also satisfies the consequent. The semantic conditions relating to axioms and ontologies are unchanged, so an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology.

4.1 Interpreting Rules

From the OWL Semantics and Abstract Syntax document [37] we recall that an abstract OWL interpretation is a tuple of the form

$$\mathcal{I} = \langle R, EC, ER, L, S, LV \rangle,$$

where R is a set of resources, $LV \subseteq R$ is a set of literal values, EC is a mapping from classes and datatypes to subsets of R and LV respectively, ER is a mapping from properties to binary relations on R, L is a mapping from typed literals to elements of LV, and S is a mapping from individual names to elements of $EC(\mathsf{owl:Thing})$.

Given an abstract OWL interpretation $\mathcal{I}$, a binding $B(\mathcal{I})$ is an abstract OWL interpretation that extends $\mathcal{I}$ such that S maps i-variables to elements of $EC(\mathsf{owl:Thing})$ and L maps d-variables to elements of LV respectively. An atom is satisfied by a binding $B(\mathcal{I})$ under the conditions given in Table 1, where C is an OWL DL description, P is an OWL DL *individual-valued* Property, Q is an OWL DL *data-valued* Property, x, y are variables or OWL individuals, and z is a variable or an OWL data value.

Atom	Condition on Interpretation
$C(x)$	$S(x) \in EC(C)$
$P(x, y)$	$\langle S(x), S(y)\rangle \in ER(P)$
$Q(x, z)$	$\langle S(x), L(z)\rangle \in ER(Q)$
sameAs(x, y)	$S(x) = S(y)$
differentFrom(x, y)	$S(x) \neq S(y)$

Table 1: Interpretation Conditions

A binding $B(\mathcal{I})$ satisfies an antecedent A iff A is empty or $B(\mathcal{I})$ satisfies every atom in A. A binding $B(\mathcal{I})$ satisfies a consequent C iff C is not empty and $B(\mathcal{I})$ satisfies every atom in C. A rule is satisfied by an interpretation $\mathcal{I}$ iff for every binding B such that $B(\mathcal{I})$ satisfies the antecedent, $B(\mathcal{I})$ also satisfies the consequent.

The semantic conditions relating to axioms and ontologies are unchanged. In particular, an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology; an ontology is consistent iff it is satisfied by at least one interpretation; an ontology O_2 is entailed by an ontology O_1 iff every interpretation that satisfies O_1 also satisfies O_2.

4.2 Example

Consider, for example, the "uncle" rule (1) from Section 3.2. Assuming that parent, brother and uncle are *individualvaluedPropertyID*s, then given an interpretation $\mathcal{I} = \langle R, EC, ER, L, S, LV\rangle$, a binding $B(\mathcal{I})$ extends S to map the variables $?a$, $?b$, and $?c$ to elements of $EC(\mathsf{owl:Thing})$; we will use a, b, and c respectively to denote these elements. The antecedent of the rule is satisfied by $B(\mathcal{I})$ iff $(a, b) \in ER(\mathsf{parent})$ and $(b, c) \in ER(\mathsf{brother})$. The consequent of the rule is satisfied by $B(\mathcal{I})$ iff $(a, c) \in ER(\mathsf{uncle})$. Thus the rule is satisfied by $\mathcal{I}$ iff for every binding $B(\mathcal{I})$ such that $(a, b) \in ER(\mathsf{parent})$ and $(b, c) \in ER(\mathsf{brother})$, then it is also the case that $(a, c) \in ER(\mathsf{uncle})$, i.e.:

$$\forall a, b, c \in EC(\mathsf{owl:Thing}).$$
$$((a, b) \in ER(\mathsf{parent}) \wedge (b, c) \in ER(\mathsf{brother})) \rightarrow (a, c) \in ER(\mathsf{uncle})$$

5 XML Concrete Syntax

Many possible XML encodings could be imagined, but the most obvious solution is to extend the existing OWL Web Ontology Language XML Presentation Syntax [17],

which can be straightforwardly modified to deal with SWRL.[3] This has several advantages:

- arbitrary OWL classes (e.g., descriptions) can be used as predicates in rules;
- rules and ontology axioms can be freely mixed;
- the existing XSLT stylesheet[4] can easily be extended to provide a mapping to RDF graphs that extends the OWL RDF/XML exchange syntax (see Section 8).

In the first place, the ontology root element is extended so that ontologies can include rule axioms and variable declarations as well as OWL axioms, import statements etc. We then simply need to add the relevant syntax for variables and rules. In this paper we use the unspecified `owlr` namespace prefix for the newly introduced syntax (the `owlx` namespace prefix, which should be treated as being bound to `http://www.w3.org/2003/05/owl-xml`, is used for the existing OWL XML syntax). In practice, the `owlr` prefix would have to be bound to some appropriate namespace name (e.g., the OWL namespace name, the OWL XML namespace name, or some new namespace name).

Variable declarations are statements about variables, indicating that the given URI is to be used as a variable, and (optionally) adding any annotations. For example:

```
<owlr:Variable owlr:name="x1" />,
```

states that the URI x1 (in the current namespace) is to be treated as a variable.

Rule axioms are similar to OWL SubClassOf axioms, except they have owlr:Rule as their element name. Like SubClassOf and other axioms they may include annotations. Rule axioms have an antecedent (owlr:antecedent) component and a consequent (owlr:consequent) component. The antecedent and consequent of a rule are both lists of atoms and are read as the conjunction of the component atoms. Atoms can be formed from unary predicates (classes), binary predicates (properties), equalities or inequalities.

Class atoms consist of a description and either an individual name or a variable name, where the description in a class atom may be a class name, or may be a complex description using boolean combinations, restrictions, etc. For example,[5]

```
<owlr:classAtom>
  <owlx:Class owlx:name="Person" />
  <owlr:Variable owlr:name="x1" />
</owlr:classAtom>
```

is a class atom using a class name (#Person), and

```
<owlr:classAtom>
  <owlx:IntersectionOf>
    <owlx:Class owlx:name="Person" />
    <owlx:ObjectRestriction
        owlx:property="hasParent">
      <owlx:someValuesFrom
```

[3]The syntax used in the W3C Member Submission was changed slightly in order to make it more compatible with RuleML (see `http://www.ruleml.org/`).

[4]`http://www.w3.org/TR/owl-xmlsyntax/owlxml2rdf.xsl`

[5]Note that we use the `owlx` namespace prefix for the names used in examples.

```
            owlx:class="Physician" />
        </owlx:ObjectRestriction>
    </owlx:IntersectionOf>
    <owlr:Variable owlr:name="x2" />
</owlr:classAtom>
```

is a class atom using a complex description representing Persons having at least one parent who is a Physician.

Property atoms consist of a property name and two elements that can be individual names, variable names or data values (as OWL does not support complex property descriptions, a property atom takes only a property name). Note that in the case where the second element is an individual name the property must be an *individual-valued* Property, and in the case where the second element is a data value the property must be a *data-valued* Property. For example:

```
<owlr:individualPropertyAtom
                    owlx:property="hasParent">
    <owlr:Variable owlr:name="x1" />
    <owlx:Individual owlx:name="John" />
</owlr:individualPropertyAtom>
```

is a property atom using an *individual-valued* Property (the second element is an individual), and

```
<owlr:datavaluedPropertyAtom owlr:property="grade">
    <owlr:Variable owlr:name="x1" />
    <owlx:DataValue
      rdf:datatype="&xsd;integer">4</owlx:DataValue>
</owlr:datavaluedPropertyAtom>
```

is a property atom using a *data-valued* Property (the second element is a data value, in this case an integer).

Finally, same (different) individual atoms assert equality (inequality) between sets of individual and variable names. Note that (in)equalities can be asserted between arbitrary combinations of variable names and individual names. For example:

```
<owlr:sameIndividualAtom>
    <owlr:Variable owlr:name="x1" />
    <owlr:Variable owlr:name="x2" />
    <owlx:Individual owlx:name="Clinton" />
    <owlx:Individual owlx:name="Bill_Clinton" />
</owlr:sameIndividualAtom>
```

asserts that the variables $x1$, $x2$ and the individual names Clinton and Bill_Clinton all refer to the same individual.

5.1 Example

The example rule from Section 3.2 can be written in the XML concrete syntax for rules as

```
<owlx:Rule>
    <owlr:antecedent>
```

```
      <owlr:individualPropertyAtom
                          owlr:property="parent">
        <owlr:Variable owlr:name="a" />
        <owlr:Variable owlr:name="b" />
      </owlr:individualPropertyAtom>
      <owlr:individualPropertyAtom
                          owlr:property="brother">
        <owlr:Variable owlr:name="b" />
        <owlr:Variable owlr:name="c" />
      </owlr:individualPropertyAtom>
    </owlr:antecedent>
    <owlr:consequent>
      <owlr:individualPropertyAtom
                          owlr:property="uncle">
        <owlr:Variable owlr:name="a" />
        <owlr:Variable owlr:name="c" />
      </owlr:individualPropertyAtom>
    </owlr:consequent>
  </owlr:Rule>
```

6 The Power of Rules

In OWL, the only relationship that can be asserted between properties is subsumption between atomic property names, e.g., asserting that hasFather is a subPropertyOf hasParent. In Section 3.2 we have already seen how a rule can be used to assert more complex relationships between properties. While this increased expressive power is clearly very useful, it is easy to show that it leads to the undecidability of key inference problems, in particular ontology consistency.

For extensions of languages such as OWL DL, the undecidability of the consistency problem is often proved by showing that the extension makes it possible to encode a known undecidable domino problem [4] as an ontology consistency problem. In particular, it is well known that such languages only need the ability to represent an infinite 2-dimensional grid in order for consistency to become undecidable [2, 24]. With the addition of rules, such an encoding is trivial. For example, given two properties x-succ and y-succ, the rule:

$$\text{x-succ}(?a, ?b) \wedge \text{y-succ}(?b, ?c) \wedge \text{y-succ}(?a, ?d) \wedge \text{x-succ}(?d, ?e) \rightarrow \text{sameAs}(?c, ?e),$$

along with the assertion that every grid node is related to exactly one other node by each of x-succ and y-succ, allows such a grid to be represented. This would be possible even without the use of the sameAs atom in the consequent—it would only be necessary to establish appropriate relationships with a "diagonal" property:

$$\text{x-succ}(?a, ?b) \wedge \text{y-succ}(?b, ?c) \rightarrow \text{diagonal}(?a, ?c)$$
$$\text{y-succ}(?a, ?d) \wedge \text{x-succ}(?d, ?e) \rightarrow \text{diagonal}(?a, ?e),$$

and additionally assert that every grid node is related to exactly one other node by diagonal.

The proposed form of OWL rules seem to go beyond basic Horn clauses in allowing:

- conjunctive consequents;
- class descriptions as well as class names as predicates in class atoms; and
- equalities and inequalities.

On closer examination, however, it becomes clear that most of this is simply "syntactic sugar", and does not add to the power of the language.

In the case of conjunctive consequents, it is easy to see that these could be eliminated by rewriting using standard rules of distributivity. For example, the rule

$$A \rightarrow C_1 \wedge C_2$$

is equivalent to $\neg A \vee (C_1 \wedge C_2)$ and, via distributivity, to $(\neg A \vee C_1) \wedge (\neg A \vee C_2)$, so can be rewritten as a semantically equivalent pair of rules

$$A \rightarrow C_1$$
$$A \rightarrow C_2.$$

In the case of class descriptions, it is easy to see that a description d can be eliminated from a rule simply by adding an OWL axiom that introduces a new class name and asserts that it is equivalent to d, e.g.,

$$\mathsf{EquivalentClasses}(D\ d).$$

The description can then be replaced with the name, here replacing the description d with class name D.

In the case of equality atoms, the sameAs property could easily be substituted with a "user defined" owl property called, for example, Eq. Such a property can be given the appropriate meaning using a rule of the form

$$\mathsf{Thing}(?x) \rightarrow \mathsf{Eq}(?x, ?x) \tag{2}$$

and by asserting that it is functional. It is easy to see that the interpretation of Eq corresponds to equality of elements in $EC(\mathsf{owl:Thing})$, i.e.,

$$\forall x, y \in EC(\mathsf{owl:Thing}).\langle x, y\rangle \in ER(\mathsf{Eq}) \iff x = y,$$

and that Eq could therefore be used instead of sameAs without changing the meaning of the ontology.

Proof: For the if direction, assume that for some interpretation $\mathcal{I}$ there exists an element x of $EC(\mathsf{owl:Thing})$ such that $\langle x, x\rangle \notin ER(\mathsf{Eq})$. Then a binding $B(\mathcal{I})$ could extend $\mathcal{I}$ so that S maps $?x$ to x, and rule 2 would not be satisfied by $B(\mathcal{I})$. For the only if direction, assume that for some interpretation $\mathcal{I}$ there exist elements x, y of $EC(\mathsf{owl:Thing})$ such that $\langle x, y\rangle \in ER(\mathsf{Eq})$ and $x \neq y$. From the if direction we also have that $\langle x, x\rangle \in ER(\mathsf{Eq})$, so Eq would not be functional.

The case of inequalities is slightly more complex. An owl property called, for example, Neq, can be introduced and used to capture some of the meaning of the differentFrom property by adding a rule of the form

$$\mathsf{Eq}(?x, ?y) \wedge \mathsf{Neq}(?x, ?y) \rightarrow \mathsf{Nothing}(?x). \tag{3}$$

It is easy to see that the interpretation of Neq is disjoint from the interpretation of Eq, i.e.,

$$\forall x, y \in EC(\mathsf{owl:Thing}).\langle x, y\rangle \in ER(\mathsf{Neq}) \implies x \neq y,$$

and that this leads to the implicit rule

$$\mathsf{Neq}(?x, ?y) \rightarrow \mathsf{differentFrom}(?x, ?y).$$

Proof: Assume that for some interpretation $\mathcal{I}$ there exist elements x, y of $EC(\mathsf{owl:Thing})$ such that $\langle x, y\rangle \in ER(\mathsf{Neq})$ and $\langle x, y\rangle \notin ER(\mathsf{differentFrom})$. If $\langle x, y\rangle \notin ER(\mathsf{differentFrom})$, then $x = y$ and $\langle x, y\rangle \in ER(\mathsf{Eq})$. A binding $B(\mathcal{I})$ could, therefore, extend $\mathcal{I}$ so that S maps $?x$ to x and $?y$ to y, and rule 3 would imply that $x \in EC(\mathsf{owl:Nothing})$, violating the semantic conditions on $\mathcal{I}$.

Rule 3 shows that we could eliminate differentFrom when it occurs in the consequent of a rule simply by substituting Neq. Neq does not, however, fully capture the meaning of inequality, because there could be pairs of elements in $EC(\mathsf{owl:Thing})$ that are in the extension of neither Eq nor Neq, i.e., differentFrom does *not* imply Neq. As a result, we cannot use Neq to eliminate occurrences of differentFrom in the antecedent of a rule: in order to do so would require Neq to be equivalent to the negation of Eq.

7 Examples of SWRL

We give two further examples of SWRL that serve to illustrate some of its utility, and show how the power of SWRL goes beyond that of either OWL DL or Horn rules alone.

7.1 Transferring Characteristics

The first example is due to Guus Schreiber, and is based on ontologies used in an image annotation demo [16].

$$\mathsf{Artist}(?x) \wedge \mathsf{Style}(?y) \wedge \mathsf{artistStyle}(?x, ?y) \wedge \mathsf{creator}(?x, ?z) \rightarrow \mathsf{style/period}(?z, ?y)$$

The rule expresses the fact that, given knowledge about the Style of certain Artists (e.g., van Gogh is an Impressionist painter), we can derive the style/period of an art object from the value of the creator of the art object, where Style is a term from the Art and Architecture Thesaurus (AAT),[6] Artist is a class from the Union List of Artist Names (ULAN),[7] artistStyle is a property relating ULAN Artists to AAT Styles, and both creator and style/period are properties from the Visual Resources Association catalogue (VRA),[8] with creator being a subproperty of the Dublin Core element dc:creator.[9]

This rule would be expressed in the XML concrete syntax as follows (assuming appropriate entity declarations):

```
<owlr:Rule>
  <owlr:antecedent>
```

[6] http://www.getty.edu/research/tools/vocabulary/aat/
[7] http://www.getty.edu/research/conducting_research/vocabularies/ulan/
[8] http://www.vraweb.org/
[9] http://dublincore.org/

```
      <owlr:classAtom>
        <owlx:Class owlx:name="&ulan;Artist" />
        <owlr:Variable owlr:name="x" />
      </owlr:classAtom>
      <owlr:classAtom>
        <owlx:Class owlx:name="&aat;Style" />
        <owlr:Variable owlr:name="y" />
      </owlr:classAtom>
      <owlr:individualPropertyAtom
                owlr:property="&aatulan;artistStyle">
        <owlr:Variable owlr:name="x" />
        <owlr:Variable owlr:name="y" />
      </owlr:individualPropertyAtom>
      <owlr:individualPropertyAtom
                owlr:property="&vra;creator">
        <owlr:Variable owlr:name="x" />
        <owlr:Variable owlr:name="z" />
      </owlr:individualPropertyAtom>
    </owlr:antecedent>
    <owlr:consequent>
      <owlr:individualPropertyAtom
                owlr:property="&vra;style/period">
        <owlr:Variable owlr:name="z" />
        <owlr:Variable owlr:name="y" />
      </owlr:individualPropertyAtom>
    </owlr:consequent>
  </owlr:Rule>
```

The example is interesting because it shows how rules can be used to "transfer characteristics" from one class of individuals to another via properties other than subClassOf—in this case, the Style characteristics of an Artist (if any) are transferred (via the creator property) to the objects that he/she creates. This idiom is much used in ontologies describing complex physical systems, such as medical terminologies, where partonomies may be as important as subsumption hierarchies, and where characteristics often need to be transfered across various partitive properties [34, 41, 44]. For example, the location of a trauma should be transfered across the partOf property, so that traumas located in a partOf an anatomical structure are also located in the structure itself [39]. This could be expressed using a rule such as

$$\mathsf{Trauma}(?x) \wedge \mathsf{Location}(?y) \wedge \mathsf{isLocatedIn}(?x, ?y) \wedge \mathsf{isPartOf}(?y, ?z) \rightarrow \mathsf{isLocatedIn}(?x, ?z)$$

A similar technique could be used to transfer properties to composite processes from their component processes when describing web services.

Terminology languages designed specifically for medical terminology such as Grail [40] and SNOMED-RT [48] often allow this kind of idiom to be expressed, but it cannot be expressed in OWL (not even in OWL full). Thus this kind of rule shows one way in which SWRL goes beyond the expressive power of OWL DL.

7.2 Inferring the Existence of New Individuals

The second example is due to Mike Dean, and illustrates a scenario in which we want to express the fact that for every Airport there is a map Point that has the same location (latitude and longitude) as the Airport and that is an object of "layer" (a map DrawingLayer).[10] Moreover, this map point has the Airport as an underlyingObject and has the Airport name as its Label. Note how the expressive power of SWRL allows "existentials" to be expressed in the head of a rule—it is asserted that, for every Airport, there must exist such a map point (using an OWL someValuesFrom restriction in a class atom). In this way SWRL goes beyond the expressive power of Horn rules.

The first part of this example is background knowledge about Airports and maps expressed in OWL DL. (A few liberties have been taken with the OWL DL abstract syntax here in the interests of better readability.) In particular, it is stated that map:location and map:object are *individual-valued* Properties with inverse properties map:isLocationOf and map:isObjectOf respectively; that latitude and longitude are *data-valued* Properties; that map:Location is a class whose instances have exactly one latitude and exactly one longitude, both being of type xsd:double; that layer is an instance of map:DrawingLayer; that map is an instance of map:Map whose map:name is `"Airports"` and whose map:layer is layer; and that airport:GEC is an instance of airport-ont:Airport whose name is `"Spokane Intl"` and whose location is latitude `47.6197` and longitude `117.5336`.

```
ObjectProperty(map:location)
ObjectProperty(map:isLocationOf
  inverseOf(map:location))
ObjectProperty(map:object)
ObjectProperty(map:isObjectOf
  inverseOf(map:location))

DatatypeProperty(latitude)
DatatypeProperty(longitude)
Class(map:Location primitive
  intersectionOf(
    restriction(latitude allValuesFrom(xsd:double))
    restriction(latitude minCardinality(1))
    restriction(longitude allValuesFrom(xsd:double))
    restriction(longitude minCardinality(1))))

Individual(layer type(map:DrawingLayer))

Individual(map type(map:Map)
  value(map:name "Airports")
  value(map:layer layer))

Individual(airport:GEC type(airport-ont:Airport)
  value(name "Spokane Intl")
  value(location Individual(value(latitude 47.6197)
          value(longitude 117.5336))))
```

[10] `http://www.daml.org/2003/06/ruletests/translation-3.n3`

The first rule in the example requires that if a map:Location is the sameLocation as another location, then it has the same values for latitude and longitude.

$$\begin{aligned}&\text{map:Location}(?maploc) \wedge \text{sameLocation}(?loc, ?maploc) \wedge \\ &\quad \text{latitude}(?loc, ?lat) \wedge \text{longitude}(?loc, ?lon) \\ &\quad\quad \rightarrow \text{latitude}(?maploc, ?lat) \wedge \text{latitude}(?maploc, ?lon)\end{aligned}$$

The second rule requires that wherever an airport-ont:Airport is located, there is some map:Location that is the sameLocation as the Airport's location, and that is the location of a map:Point that is an object of the map:DrawingLayer "layer". Note that the head of the rule is an atom of the form $C(?\text{loc})$, where the class C is an OWL restriction.

$$\begin{aligned}&\text{airport-ont:Airport}(?airport) \wedge \text{location}(?airport, ?loc) \wedge \\ &\quad \text{latitude}(?loc, ?lat) \wedge \text{longitude}(?loc, ?lon) \\ &\quad \rightarrow \text{restriction}(\text{sameLocation} \\ &\qquad \text{someValuesFrom}(\\ &\qquad\quad \text{intersectionOf}(\text{map : Location} \\ &\qquad\quad\quad \text{restriction}(\text{isLocationOf} \\ &\qquad\quad\quad\quad \text{someValuesFrom}(\\ &\qquad\quad\quad\quad\quad \text{intersectionOf}(\text{map : Point} \\ &\qquad\quad\quad\quad\quad\quad \text{restriction}(\text{map : isObjectOf} \\ &\qquad\quad\quad\quad\quad\quad\quad \text{someValuesFrom}(\text{OneOf}(\text{layer}))))))))))(?loc)\end{aligned}$$

The third rule requires that the map:Point whose map:location is the map:Location of an airport-ont:Airport has the airport as a map:underlyingObject and has a map:label which is the name of the airport.

$$\begin{aligned}&\text{airport-ont:Airport}(?airport) \wedge \text{map:location}(?airport, ?loc) \wedge \\ &\quad \text{sameLocation}(?loc, ?maploc) \wedge \text{map:Location}(?point, ?maploc) \wedge \\ &\quad\quad \text{airport-ont:name}(?airport, ?name) \\ &\quad \rightarrow \text{map:underlyingObject}(?point, ?airport) \wedge \\ &\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{map:label}(?point, ?name)\end{aligned}$$

8 Mapping to RDF Graphs

It is widely assumed that the Semantic Web will be based on a hierarchy of (increasingly expressive) languages, with RDF/XML providing the syntactic and semantic foundation (see, e.g., [5]). In accordance with this design philosophy, the charter of the W3C Web Ontology Working Group (the developers of the OWL language) explicitly stated that "*The language will use the XML syntax and datatypes wherever possible, and will be designed for maximum compatibility with XML and RDF language conventions.*". In pursuance of this goal, the working group devoted a great deal of effort to developing an RDF based syntax for OWL that was also consistent with the semantics of RDF [20]. It is, therefore, worth considering how this design might be extended to encompass rules.

One rather serious problem is that, unlike OWL, rules have variables, so treating them as a semantic extension of RDF is very difficult. It is, however, still possible

to provide an RDF syntax for rules—it is just that the semantics of the resultant RDF graphs may not be an extension of the RDF Semantics [15].

A mapping to RDF/XML is most easily created as an extension to the XSLT transformation for the OWL XML Presentation syntax.[11] This would introduce RDF classes for SWRL atoms and variables, and RDF properties to link atoms to their predicates (classes and properties) and arguments (variables, individuals or data values).[12] The example rule given in Section 7.1 (that equates the style/period of art objects with the style of the artist that created them) would be mapped into RDF as follows:

```
<owlr:Variable rdf:ID="x"/>
<owlr:Variable rdf:ID="y"/>
<owlr:Variable rdf:ID="z"/>
<owlr:Rule>
  <owlr:antecedent rdf:parseType="Collection">
    <owlr:classAtom>
      <owlr:classPredicate
        rdf:resource="&ulan;Artist"/>
      <owlr:argument1 rdf:resource="#x" />
    </owlr:classAtom>
    <owlr:classAtom>
      <owlr:classPredicate
        rdf:resource="&aat;Style"/>
      <owlr:argument1 rdf:resource="#y" />
    </owlr:classAtom>
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate
        rdf:resource="&aatulan;artistStyle"/>
      <owlr:argument1 rdf:resource="#x" />
      <owlr:argument2 rdf:resource="#y" />
    </owlr:individualPropertyAtom>
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate
        rdf:resource="&vra;creator"/>
      <owlr:argument1 rdf:resource="#x" />
      <owlr:argument2 rdf:resource="#z" />
    </owlr:individualPropertyAtom>
  </owlr:antecedent>
  <owlr:consequent rdf:parseType="Collection">
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate
        rdf:resource="&vra;style/period"/>
      <owlr:argument1 rdf:resource="#z" />
      <owlr:argument2 rdf:resource="#y" />
    </owlr:individualPropertyAtom>
  </owlr:consequent>
</owlr:Rule>
```

where &ulan;, &aat;, &aatulan; and &vra; are assumed to expand into the appropriate

[11] http://www.w3.org/TR/owl-xmlsyntax/owlxml2rdf.xsl

[12] The result is similar to the RDF syntax for representing disjunction and quantifiers proposed in [30].

namespace names. Note that complex OWL classes (such as OWL restrictions) as well as class names can be used as the object of SWRL's classPredicate property.

9 Reasoning Support for SWRL

Although SWRL provides a fairly minimal rule extension to OWL, the consistency problem for SWRL ontologies is still undecidable (as we have seen in Section 6). This raises the question of how reasoning support for SWRL might be provided.

It seems likely, at least in the first instance, that many implementations will provide only partial support for SWRL. For this reason, users may want to restrict the form or expressiveness of the rules and/or axioms they employ either to fit within a tractable or decidable fragment of SWRL, or so that their SWRL ontologies can be handled by existing or interim implementations.

One possible restriction in the form of the rules is to limit antecedent and consequent classAtoms to be named classes, with OWL axioms being used to assert additional constraints on the instances of these classes (in the same document or in external OWL documents). Adhering to this format should make it easier to translate rules to or from existing (or future) rule systems, including Prolog, production rules (descended from OPS5), event-condition-action rules and SQL (where views, queries, and facts can all be seen as rules); it may also make it easier to extend existing rule based reasoners for OWL (such as Euler[13] or FOWL[14]) to handle SWRL ontologies. Further, such a restriction would maximise backwards compatibility with OWL-speaking systems that do not support SWRL. It should be pointed out, however, that there may be some incompatibility between the first order semantics of SWRL and the Herbrand model semantics of many rule based reasoners.

By further restricting the form of rules and DL axioms used in SWRL ontologies it would be possible to stay within DLP, a subset of the language that has been shown to be expressible in either OWL DL or declarative logic programs (LP) alone [14]. This would allow either OWL DL reasoners or LP reasoners to be used with such ontologies, although there may again be some incompatibility between the semantics of SWRL and those of LP reasoners.

Another obvious strategy would be to restrict the form of rules and DL axioms so that a "hybrid" system could be used to reason about the resulting ontology. This approach has been used, e.g., in the CLASSIC [38] and CARIN systems [28], where sound and complete reasoning is made possible mainly by focusing on query answering, by restricting the DL axioms to languages that are *much* weaker than OWL, by restricting the use of DL terms in rules, and/or by giving a different semantic treatment to rules.

Finally, an alternative way to provide reasoning support for SWRL would be to extend the translation of OWL into TPTP[15] implemented in the Hoolet system,[16] and use a first order prover such as Vampire to reason with the resulting first order theory [42, 54]. This technique would have several advantages: no restrictions on the form of SWRL rules or axioms would be required; the use of a first order prover would ensure that all inferences were sound with respect to SWRL's first order semantics; and the use of the TPTP syntax would make it possible to use any one of a range of state of the

[13] `http://www.agfa.com/w3c/euler/`

[14] `http://fowl.sourceforge.net`

[15] A standard syntax used by many first order theorem provers—see `http://www.tptp.org`

[16] `http://www.w3.org/2003/08/owl-systems/test-results-out`

art first order provers. A prototype based on this approach is described in the following section.

10 A Prototype SWRL Reasoner

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) [10, 19, 1]. This suggests the idea of using standard methods of automated reasoning for FOL as a mechanism for reasoning with OWL DL.

This might be done by trying to create from scratch new architectures for reasoning in 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. An additional attraction of using a FO prover in this way is the fact that the translation from DL to FOL can be extended to handle SWRL, providing an implementation of a SWRL reasoner.

Here we describe our initial prototype implementation of just such a SWRL reasoner, known as Hoolet. It should be noted that this initial implementation is rather simplistic, and is only intended as a preliminary feasibility study. We will, however, discuss the issue of possible optimisations.

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 [36], while Ganzinger and de Nivelle have developed decision procedures for the guarded fragment, a fragment of FOL that includes many description logics [11]. The most widely known work in this area was by Hustadt and Schmidt [26], who used the SPASS FOL prover to reason with propositional modal logics, and, via well known correspondences [45], 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 [18], and it is not even clear if such an extension would be possible.

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

In recent years, a number of highly efficient FO provers have been implemented [32, 50, 43]. These provers compete annually on a set of tasks, and the results are published [9]. One of the most successful general-purpose provers has been Vampire [43], and we have chosen this prover to use in our prototype.

Vampire is a general-purpose FOL prover developed by Andrei Voronkov and Alexandre Riazanov. Given a set of first-order formulas, 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, 33]). 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[17] has been used for saturation.

The main input format of Vampire is the TPTP syntax [49] (although a parser for a subset of KIF [12] has been added recently). Using the TPTP syntax in our prototype means that it would be possible to substitute Vampire with any one of a range of state of the art first order provers.

10.1 Translation issues

Translating OWL Ontologies into FOL Axioms 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 [19]. The translation ϕ maps DL concepts C and role names 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 [7, 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 [31]. 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 [13].

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 ($\mathcal{RIQ}$ [22]),

[17] I.e., all generated clauses are eventually processed

may lead to the introduction of three or more variables.[18] The above transitivity axiom for role T is an example of this case. FOL with three variables is known to be undecidable [7].

OWL DL also provides for XML schema *datatypes* [6], equivalent to a very simple form of *concrete domains* [21]. 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 [37].

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) [37].

Translating SWRL Rules into FOL Axioms Using the translation approach, we can easily extend the first-order translation to SWRL rules and thus provide a simple implementation of a SWRL reasoner.

As we have seen, 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 2, where C is an OWL class description, R an OWL property and i and j are either OWL individual names or SWRL variables.

Table 2: Rule Atoms

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

In our prototype we have only considered a simplification of SWRL 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 expressiveness 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 Section 4. Each rule is translated as an implication, and any free variables in the rule are

[18] In some cases, the effects of transitive roles can be axiomatised in $\mathcal{C}^2$ [53].

assumed to be universally quantified. Thus a rule:

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

is translated to an axiom:

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

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 3. Combining this translation with the translation from OWL to FOL described above 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 (Vampire in our case), where it can be used for reasoning tasks such as satisfiability checking and instance checking.

Table 3: Rule Atom Translation

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

10.2 Examples

As an example, we will consider a variant on the "uncle" example given in Section 3.2:

```
hasParent(?x,?y), hasSibling(?y, ?z), Male(?z)
    ⇒ 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 `hasUncle`(`Robert`, `Ian`), but also that `Ian` is an instance of the `Uncle` class.

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

```
Beer(?x) ⇒ 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.

10.3 Performance and Optimisation

Our prototype works well with small examples, such as those given in Sections 7 and 10.2, and we have used it successfully with SWRL ontologies containing up to 100 axioms, rules and facts. However, while it is useful to have a prototype that can be used for illustrative and test purposes, the effectiveness of such a naive approach must be open to question with larger SWRL ontologies.

In [55] it was shown that, when using the same translation approach to reason with OWL DL ontologies, performance could be greatly improved by using a so-called "relevant only" translation. The key idea is that when ontologies are translated, Vampire receives *all* of the axioms that occur in the ontology, whereas usually only a small fraction of them are actually relevant to a given subsumption or inconsistency problem. Vampire is not optimised to deal efficiently with large numbers of irrelevant axioms, and so it does not perform well under these circumstances.

An obvious way to correct this situation is to remove all irrelevant information from the FO task given to Vampire. An axiom is said to be *irrelevant* to a consistency test of C if it can easily be shown (i.e., via a syntactic analysis) that removing it from the ontology would not affect the interpretation of C; other axioms are called *relevant*. Note that not every "relevant axiom" really *will* affect the computation of the consistency of C, but we cannot (easily) rule out the possibility that it *may* affect the computation. An FO-translation is called *relevant-only* if it contains only FO-translations of axioms relevant (in the above sense) to the given satisfiability test.

The definition of relevance given in [55] can be extended to SWRL by treating rules in the same way as general concept inclusion axioms (GCIs). 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 and rule in the ontology.[19] *Relevance* is the transitive closure of depends. The process of selecting information relevant to a concept expression E looks very much the same as *unfolding* (see [1]), and assumes that the KB is separated into a set of unfoldable axioms and a set of GCIs [25] and rules. Every concept name CN and role name RN appearing in E is relevant to E. The process is then repeated recursively for unfoldable axioms with CN on the left hand side (whether inclusion or equality axioms). Also, if role R is relevant to E, then so are all roles R' s.t. $R \sqsubseteq R'$, along with their inverses (if the target DL allows inverse roles). An algorithm for computing relevant information is quite straightforward and is described in detail in [54].

Computing relevance leads to a small overhead when translating a SWRL ontology into FOL, but it should greatly increase the performance of the FO prover. Preliminary experiments with an extension of Hoolet to include an implementation of the relevant only translation suggest that this is indeed the case [29].

11 Discussion

In this paper we have presented SWRL, a proposed extension to OWL to include a simple form of Horn-style rules. We have provided formal syntax and semantics for SWRL, shown how OWL's XML and RDF syntax can be extended to deal with SWRL,

[19] It should be possible to treat (some) rules as unfoldable axioms, add thus eliminate the need to include all rules in the relevant only translation, but this is still the subject of ongoing work.

illustrated the features of SWRL with several examples, and discussed how reasoning support for SWRL might be provided.

The main strengths of the proposal are its simplicity and its tight integration with the existing OWL language. As we have seen, SWRL extends OWL with the most basic kind of Horn rule (sweetened with a little "syntactic sugar"): predicates are limited to being OWL classes and properties (and so have a maximum arity of 2), there are no disjunctions or negations (of atoms), no built in predicates (such as arithmetic predicates), and no nonmonotonic features such as negation as failure or defaults. Moreover, rules are given a standard first order semantics. This facilitates the tight integration with OWL, with SWRL being defined as a syntactic and semantic extension of OWL DL.

While we believe that SWRL defines a natural and useful level in the hierarchy of Semantic Web languages, it is clear that some applications would benefit from further extensions in expressive power. In particular, the ability to express arithmetic relationships between data values is important in many applications (e.g., to assert that persons whose income at least equals their expenditure are happy, while those whose expenditure exceeds their income are unhappy). It is not clear, however, if this would best be achieved by extending SWRL to include rules with built in arithmetic predicates, or by extending OWL Datatypes to include nary predicates [35].

Finally, we have shown how a first order theorem prover can be used to provide reasoning services for SWRL, and how some simple optimisations can be used to improve performance. Our results were sufficiently encouraging to suggest that, with further tuning and optimisation, such a strategy would be useful in (some) realistic applications. Future work will include such tuning and optimisation, as well as empirical investigations to determine the practical value of the resulting system.

Acknowledgements

This document has benefited from extensive discussion in the Joint US/EU ad hoc Agent Markup Language Committee. Parts of Section 9, in particular, were the result of feedback from and discussions with Benjamin Grosof and Mike Dean.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002.

[2] F. Baader and U. Sattler. Number restrictions on complex roles in description logics: A preliminary report. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 328–338, 1996.

[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] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.

[5] T. Berners-Lee. Semantic web roadmap, 1998. Available at `http://www.w3.org/DesignIssues/Semantic`.

[6] P. V. Biron and A. Malhotra. XML schema part 2: Datatypes. W3C Recommendation, May 2001. http://www.w3.org/TR/xmlschema-2/.

[7] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[8] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.

[9] The CASC web page. http://www.cs.miami.edu/~tptp/CASC/.

[10] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-ref/.

[11] H. Ganzinger and H. 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.

[12] 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.

[13] E. Gradel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proceedings of LICS 1997*, pages 306–317, 1997.

[14] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 48–57. ACM, 2003.

[15] P. Hayes. RDF model theory. W3C Recommendation, 10 February 2004. Available at http://www.w3.org/TR/rdf-mt/.

[16] L. Hollink, G. Schreiber, J. Wielemaker, and B. Wielinga. Semantic annotation of image collections. In *Workshop on Knowledge Markup and Semantic Annotation, KCAP'03*, 2003. Available at http://www.cs.vu.nl/~guus/papers/Hollink03c.pdf.

[17] M. Hori, J. Euzenat, and P. F. Patel-Schneider. OWL web ontology language XML presentation syntax. W3C Note, 11 June 2003. Available at http://www.w3.org/TR/owl-xmlsyntax/.

[18] I. Horrocks and P. 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.

[19] I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In D. Fensel, K. Sycara, and J. 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.

[20] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[21] I. Horrocks and U. 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.

[22] I. Horrocks and U. 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.

[23] I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 160(1–2):79–104, Dec. 2004.

[24] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer, 1999.

[25] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 285–296, 2000.

[26] 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.

[27] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-concepts/`.

[28] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[29] K. Li. Optimising first order reasoning over owl ontologies. Master's thesis, University of Manchester, 2004.

[30] D. V. McDermott and D. Dou. Representing disjunction and quantifiers in rdf. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 250–263. Springer, 2002.

[31] M. Mortimer. On languages with two variables. *Zeitschr. für math. Logik und Grundlagen der Math.*, 21:135–140, 1975.

[32] M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schumann, and K. Mayr. SETHEO and e-SETHEO - the CADE-13 systems. *Journal of Automated Reasoning*, 18(2):237–246, 1997.

[33] 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.

[34] L. Padgham and P. Lambrix. A framework for part-of hierarchies in terminological logics. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 485–496, 1994.

[35] J. Pan and I. Horrocks. Web ontology reasoning with datatype groups. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 47–63. Springer, 2003.

[36] M. Paramasivam and D. A. Plaisted. Automated deduction techniques for classification in description logic systems. *J. of Automated Reasoning*, 20(3):337–364, 1998.

[37] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[38] P. F. Patel-Schneider, D. L. McGuiness, R. J. Brachman, L. A. Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bull.*, 2(3):108–113, 1991.

[39] A. Rector. Analysis of propagation along transitive roles: Formalisation of the galen experience with medical ontologies. In *Proc. of DL 2002*. CEUR (`http://ceur-ws.org/`), 2002.

[40] A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.

[41] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, 1997.

[42] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[43] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[44] U. Sattler. Description logics for the representation of aggregated objects. In *Proc. of ECAI 2000*. IOS Press, 2000.

[45] K. 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.

[46] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431. Morgan Kaufmann, Los Altos, 1989.

[47] M. K. Smith, C. Welty, and D. L. McGuinness. OWL web ontology language guide. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/owl-guide/`.

[48] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. *J. of the Amer. Med. Informatics Ass.*, 2000. Fall Symposium Special Issue.

[49] G. Sutcliffe and C. Suttner. The TPTP Problem Library. TPTP v. 2.4.1. Technical report, University of Miami, 2001.

[50] T. Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199–204, 1997.

[51] The DAML Services Coalition. Daml-s: Semantic markup for web services, May 2003. Available at `http://www.daml.org/services/daml-s/0.9/daml-s.html`.

[52] Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, 6 October 2000. Available at `http://www.w3.org/TR/REC-xml`.

[53] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[54] D. Tsarkov and I. 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.

[55] D. Tsarkov, A. Riazanov, S. Bechhofer, and I. Horrocks. Using Vampire to reason with OWL. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, number 3298 in Lecture Notes in Computer Science, pages 471–485. Springer, 2004.