A Proposal for an OWL Rules Language

Ian Horrocks
University of Manchester
Manchester, UK
horrocks@cs.man.ac.uk

Peter F. Patel-Schneider
Bell Labs Research
Murray Hill, NJ, USA
pfps@research.bell-labs.com

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 ORL (OWL Rules Language), a Horn clause rules extension to OWL that overcomes many of these limitations. ORL extends OWL in a syntactically and semantically coherent manner: the basic syntax for ORL rules is an extension of the abstract syntax for OWL DL and OWL Lite; ORL rules are given formal meaning via an extension of the OWL DL model-theoretic semantics; ORL rules are given an XML syntax based on the OWL XML presentation syntax; and a mapping from ORL rules to RDF graphs is given based on the OWL RDF/XML exchange syntax. We discuss the expressive power of ORL, showing that the ontology consistency problem is undecidable, provide several examples of ORL usage, and discuss how reasoning support for ORL might be provided.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages; F.4.1 [**Mathematical Logic**]: Model theory.

General Terms

Languages, Theory

Keywords

Semantic Web, representation, model-theoretic semantics

1. INTRODUCTION

The OWL Web Ontology Language [23] adds considerable expressive power to the Semantic Web. However, for a variety of reasons (see `http://lists.w3.org/Archives/Public/www-webont-wg/`), 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 [25], or in medical informatics, where it may be necessary to transfer characteristics across partitive properties [17].

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 [7]. 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" (see, e.g., [3]). 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 [15] 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, which we will refer to as OWL Rules Language (ORL), 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 ORL 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 Section 9 we discuss how reasoning support for ORL might be provided. Finally (in Section 10), we summarise the main features of the ORL proposal and suggest some directions for future work.

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

Copyright is held by the author/owner(s).
WWW2004, May 17–22, 2004, New York, NY USA.
ACM xxx.xxx.

the OWL abstract syntax and the direct model-theoretic semantics for OWL DL [15] 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 transformed (via the Lloyd-Topor transformations [11]) 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 [22] and make it easy to encode known undecidable problems as an ORL ontology consistency problem (see Section 6).

3. ABSTRACT SYNTAX

The syntax for ORL 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 [15].

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 [26]. In this notations, terminals are quoted; non-terminals are bold and 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 [9]. 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 [15].

3.1 Rules

From the OWL Semantics and Abstract Syntax document [15], 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 an ORL 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 rdf: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 transformed (via the Lloyd-Topor transformations [11] 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 individualvalued property, Q is an OWL DL datavalued 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"). As we will see in Section 6, this safety condition does not, in fact, restrict the expressive power of the language (because existentials can already be captured using OWL someValuesFrom restrictions).

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 serial-

isations, 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.

4. DIRECT MODEL-THEORETIC SEMANTICS

The model-theoretic semantics for ORL is a straightforward extension of the semantics for OWL DL given in [15]. 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 [15] 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 individualvalued property, Q is an OWL DL datavalued 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 individualvaluedPropertyIDs, 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.:

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

5. XML CONCRETE SYNTAX

Many possible XML encodings could be imagined (e.g., a RuleML based syntax as proposed in `http://www.daml.org/listarchive/joint-committee/1460.html`), but the most obvious solution is to extend the existing OWL Web Ontology Language XML Presentation Syntax [6], which can be straightforwardly modified to deal with ORL. 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[1] 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 document we use the unspecified `owlr` namespace prefix. This prefix would have to be bound to some appropriate namespace name, either the OWL 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

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

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,

```
<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
            owlx:property="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 datavalued 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 individualvalued 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 datavalued 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 [2] 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 [1, 8]. With the addition of rules, such an encoding is trivial. For example, given two properties x-succ and y-succ, the rule:

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

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:

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

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 using the standard Lloyd-Topor transformation [11]. For example, a rule of the form

$$A \rightarrow C_1 \wedge C_2$$

can be transformed into 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 predicate 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

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

and by asserting that it is functional.

The case of inequalities is slightly more complex. When they occur in the consequent of a rule they can easily be eliminated. For example, the atom

$$\mathsf{differentFrom}(x, y),$$

where x and y are again variables or constants, can be replaced with

$$C(x) \wedge D(y),$$

where C and D are new class names that are asserted to be disjoint. When (in)equalities occur in antecedents, however, this elimination does not work, because it would strengthen the conditions that must be met in order for a binding to satisfy the antecedent.

7. EXAMPLES OF ORL

We give two further examples of ORL that serve to illustrate some of their utility, and show how the power of ORL 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 [5].

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

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),[2] Artist is a class from the Union List of Artist Names (ULAN),[3] 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),[4] with creator being a subproperty of the Dublin Core element dc:creator.[5]

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

```
<owlr:Rule>
  <owlr:antecedent>
    <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 characteristic is transfered from Artists to the objects that they create via the creator property. 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 [13, 19, 21]. 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 [17]. This could be expressed using a rule such as

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

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 [18] and SNOMED-RT [24] often allow this

[2] http://www.getty.edu/research/tools/vocabulary/aat/

[3] http://www.getty.edu/research/conducting_research/vocabularies/ulan/

[4] http://www.vraweb.org/

[5] http://dublincore.org/

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 ORL go 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).[6] Moreover, this map point has the Airport as an underlyingObject and has the Airport name as its Label. Note how the expressive power of ORL 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 ORL 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 individualvalued properties with inverse properties map:isLocationOf and map:isObjectOf respectively; that latitude and longitude are datavalued 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))))
```

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.

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

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".

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

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.

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

8. MAPPING TO RDF GRAPHS

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 [4].

A mapping to RDF/XML is most easily created as an extension to the XSLT transformation for the OWL XML Presentation syntax.[7] This would introduce RDF classes for ORL atoms and variables, and RDF properties to link atoms to their predicates (classes and properties) and arguments (variables, individuals or data values).[8] 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"/>
```

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

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

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

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

9. REASONING SUPPORT FOR ORL

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

It seems likely, at least in the first instance, that many implementations will provide only partial support for ORL. 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 ORL, or so that their ORL 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[9] or FOWL[10]) to handle ORL ontologies. Further, such a restriction would maximise backwards compatibility with OWL-speaking systems that do not support ORL. It should be pointed out, however, that there may be some incompatibility between the first order semantics of ORL and the Herbrand model semantics of many rule based reasoners.

[9] `http://www.agfa.com/w3c/euler/`
[10] `http://fowl.sourceforge.net`

By further restricting the form of rules and DL axioms used in ORL 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 [3]. 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 ORL 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 [16] and CARIN systems [10], where sound and complete reasoning is made possible mainly by focusing on query answering, and by restricting the DL axioms to languages than are *much* weaker than OWL.

Finally, an alternative way to provide reasoning support for ORL would be to extend the translation of OWL into TPTP[11] implemented in the Hoolet system,[12] and use a first order prover such as Vampire to reason with the resulting first order theory [20, 27]. This technique would have several advantages: no restrictions on the form of ORL rules or axioms would be required; the use of a first order prover would ensure that all inferences were sound with respect to ORL'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 art first order provers.

10. DISCUSSION

In this paper we have presented ORL, a proposed extension to OWL to include a simple form of Horn-style rules. We have provided formal syntax and semantics for ORL, shown how OWL's XML and RDF syntax can be extended to deal with ORL, illustrated the features of ORL with several examples, and discussed how reasoning support for ORL 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, ORL 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 ORL being defined as a syntactic and semantic extension of OWL DL.

While we believe that ORL 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 ORL to include rules with built in arithmetic predicates, or by extending OWL Datatypes to include nary predicates [14].

Acknowledgements

This document has benefited from extensive discussion in the Joint US/EU ad hoc Agent Markup Language Committee. Parts of Sec-

[11] A standard syntax used by many first order theorem provers—see `http://www.tptp.org`
[12] `http://www.w3.org/2003/08/owl-systems/test-results-out`

tion 9, in particular, were the result of feedback from and discussions with Benjamin Grosof and Mike Dean.

11. REFERENCES

[1] 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.

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

[3] 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.

[4] P. Hayes. RDF model theory. W3C Working Draft, 10 October 2003. Available at `http://www.w3.org/TR/rdf-mt/`.

[5] 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`.

[6] 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/`.

[7] 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.

[8] 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.

[9] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Working Draft, 10 October 2003. Available at `http://www.w3.org/TR/rdf-concepts/`.

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

[11] J. W. Lloyd. *Foundations of logic programming (second, extended edition)*. Springer series in symbolic computation. Springer-Verlag, New York, 1987.

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

[13] 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.

[14] 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.

[15] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. W3C Candidate Recommendation, 18 August 2003. Available at `http://www.w3.org/TR/owl-semantics/`.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] M. K. Smith, C. Welty, and D. L. McGuinness. OWL web ontology language guide. W3C Candidate Recommendation, 18 August 2003. Available at `http://www.w3.org/TR/owl-guide/`.

[24] 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.

[25] 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`.

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

[27] 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.