Constructing RuleML-Based Domain Theories on top of OWL Ontologies

Christopher J. Matheus[1], Mitch M. Kokar[2],
Kenneth Baclawski[2], and Jerzy Letkowski[3]

[1] Versatile Information Systems, Inc.
Framingham, MA, USA
cmatheus@vistology.com
http://www.vistology.com
[2] Northeastern University
Boston, MA, USA
kokar@coe.neu.edu, ken@baclawski.com
[3] Western New England College
Springfield, MA, USA
jerzy@letkowski.name

Abstract. Situation Awareness involves the comprehension of the state of a collection of objects in an evolving environment. This not only includes an understanding of the objects' characteristics but also an awareness of the significant relations that hold among the objects at any point in time. Systems for establishing situation awareness require a knowledge representation for these objects and relations. Traditional *ontologies,* as defined with a language like DAML/OWL, are commonly used for such purposes. Unfortunately, these languages are insufficient for describing the conditions under which specific relations might hold true, which requires the explicit representation of *implications,* as is provided by RuleML. This paper describes an approach to knowledge representation for situation awareness employing RuleML-based domain theories constructed over OWL ontologies, presented in the context of its implementation in a Situation Awareness Assistant under development by the authors. Suggestions are also made for additions to the RuleML specification.

1 Introduction

Maintaining a coherent *situation awareness* (SAW) with respect to all relevant entities residing in a region of interest is essential for achieving successful resolution of an evolving situation [1], [2], [3]. Examples of areas where this capability is critical include air traffic control, financial markets, military battlefields and disaster management. In any situation the primary basis for SAW is knowledge of the objects within the region of interest, typically provided by "sensors" (both mechanical and human) that perform object identification and characterization; this is known as Level 1 processing in military parlance [4]. Although knowledge of the existence and attributes of individual objects is essential, full awareness also requires knowing the relations among the objects that are relevant to the current operation. For example, sim-

ply knowing that there is a west-bound airline and an east-bound airline on the radar screen is not as important as knowing that the two planes are "dangerously close" to one another. In this case, "dangerously close" is a relation between two objects that must be derived from sensory data, although the data by itself says nothing about the concepts of "closeness" or "dangerous".

Deriving relevant relations is at the core of what is referred to as Level 2 processing. Unfortunately, the problem of finding all relevant relations is a more difficult problem than merely determining the objects and their characteristics present in a situation. While the number of objects and their attributes may be large, the number scales linearly with the cardinality of the objects in the region. The same cannot be said for relations, whose possibilities increase exponentially as the number of objects increases. Furthermore, relations are abstract semantic concepts that can be constructed at will, unlike physical objects that are provided and fixed by the environment. Yet for any given situation only a small subset of all possible relations will be relevant and meaningful to the goals of the individuals who are analyzing and attempting to establish an awareness of what is occurring in the situation. It is therefore paramount that systems designed to perform Level 2 processing have a notion of the goals of the users and some knowledge about the relations that are relevant to those goals.

Systems that assist in SAW require the ability to represent objects and maintain information about their attributes and relationships with other objects as they evolve over time. This calls for the selection of some form of *data representation*. In addition, because the number of possible relations in a situation makes the problem intractable, some form of domain knowledge is required to help reduce the complexity. This necessitates the selection of some form of *knowledge representation* (KR). The domain knowledge that is required for SAW is of two types: 1) knowledge about what classes or objects, attributes and relations are possibly relevant and 2) what conditions must exist among the objects and their attributes for a given relation to hold true. Rather than selecting a single KR approach to achieve both of these requirements at once, we propose the use of OWL ontologies for the first requirement – a choice that also provides a data representation in terms of instance annotations – and RuleML rules constructed on top of these ontologies to satisfy the second requirement.

In earlier work we developed a formalization of SAW that required the generation of a SAW Core ontology [5], [6], [7], [8]. This ontology was originally developed in UML and then converted to DAML and Slang [9] for formal reasoning on SAW using SNARK [10]. We are now undertaking the development of a functional prototype Situation Awareness Assistant (SAWA) based on this formalization [11]. The SAW Core ontology at the heart of the system will be based in OWL [12]. RuleML [13] will serve as the language for defining domain theories (i.e., collections of axioms or rules). In this paper we describe how we use RuleML in conjunction with our OWL-based ontologies. We start out by discussing the role of ontologies in SAW. We then present our core SAW ontology and demonstration how it can be extended to a specific domain. This leads into a discussion of the need for rules and our selection of RuleML for this purpose. With a simple example we show how rules can be built on top of our core and domain specific ontologies using class URIs as name references. We conclude with several suggestions for additions to the RuleML specification.

2 Ontologies for Situation Awareness

SAW systems need to receive and represent situation-specific information in a computer-compatible form, which presumes the selection or creation of a descriptive language. Because we also intend to reason about this information we need a way to represent the semantics of the language such that reasoning algorithms (e.g., theorem provers) can make use of the data and knowledge representations. An effective way to achieve this is with ontologies. An *ontology* is an explicit, machine-readable semantic model of a domain that defines classes of entities along with the possible inter-class relations, called properties, specific to that domain.

As part of its Semantic Web effort, the W3C has been engaging in the development of a new XML-based language called the Web Ontology Language (OWL) [12]. OWL is an emerging standard for ontologies and knowledge representations, based on the Resource Description Framework (RDF) [14] and the DARPA Agent Markup Language (DAML), which is the immediate predecessor of OWL. OWL is a declarative, formally defined language that fully supports specialization/generalization hierarchies as well as arbitrary many-to-many relationships. Both model theoretic and axiomatic semantics have been fully defined for the elements in OWL/DAML providing strong theoretical as well as practical benefits in terms of being able to precisely define what can and cannot be achieved with these languages [15]. The field is relatively young, yet several tools have been developed and many more are on the horizon for creating OWL ontologies and processing OWL documents (for a review of such tools see [16]). In our work, we create ontologies as UML diagrams and then programmatically convert them into DAML/OWL representations [17].

It should be noted that the job performed by ontology languages such as OWL cannot be accomplished with purely syntactic languages such as XML Schema. An XML Schema specification can define the structure of objects (*i.e.,* their composition) but it cannot capture the semantic meaning implicit in the relations that might exist between classes of objects. For example, it is not possible to state in XML Schema that the meaning of <address> in one part of a document is of the same class (and thus has the same meaning) as the tag <place> used in another part of the same or different documents. To do this requires the ability to represent knowledge about how classes of objects are related to one another, which is precisely what ontologies capture.

Once an ontology is constructed it can be used to create *instance annotations.* Instance annotations are descriptions of collections of objects marked up in terms of the classes and properties (which define inter-class relations) of an ontology. We will provide some examples of instance annotations after we described our SAW ontology and its extension to a specific domain.

2.1 A SAW Core Ontology

Our original work with SAW involved the development of a formalization of SAW [5,6,7,8] which consisted of a formal definition of SAW, a core SAW ontology, a methodology for reasoning about SAW and the realization of all three as sorts, ops and axioms in Specware [9]. For this paper we will concentrate on our SAW Core ontology, which is depicted in Fig. 1. Shown is a UML diagram of the ontology in

which rectangles represent classes and connecting lines indicate inter-class relationships or properties. In the rest of this section we provide an overview of the most important classes and relationships; for a more detailed discussion of our SAW Ontology along with alternative design considerations see [7].

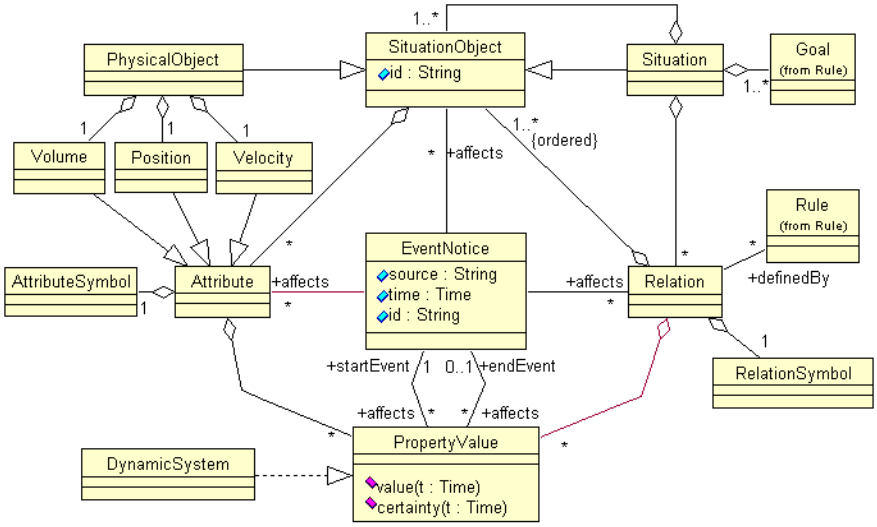

Fig. 1. Core SAW Ontology

The *Situation* class (upper right corner) defines a situation to be a collection of Goals, SituationObjects and Relations. *SituationObjects* are entities in a situation -- both physical and abstract -- that can have characteristics (*i.e.,* Attributes) and can participate in relationships with other objects (*i.e.,* Relations). *Attributes* define values of specific object characteristics, such as position, weight or color. A *PhysicalObject* is a special type of SituationObject that necessarily has the attributes of Volume, Position and Velocity. *Relations* define the relationships among ordered sets of SituationObjects.

An important aspect of Attributes and Relations is that they need to be associated with values that can change over time. To accomplish this Attributes/Relations are associated with zero or more *PropertyValues* each of which defines two time dependant functions, one for the actual *value* and the other for the *certainty* assigned to that value. A new PropertyValue is created for an Attribute/Relation whenever an EventNotice arrives that "affects" that Attribute/Relation. The value of an Attribute/Relation at a particular point in time (either current, past or future) can be determined by accessing the value function of the PropertyValue instance that is in effect at the prescribed time. This is illustrated in Fig. 2, but before explaining the illustration we need to introduce the EventNotice class.

EventNotices contain information about events in the real-world situation as observed by a sensor (the *source)* at a specific *time* that *affects* a specific Relation or At-

tribute (of a specific SituationObject*)* by defining or constraining its PropertyValue. These are the entities that indicate change in the situation and thus are the vehicles by which changes are affected in the Attributes and Relations of the situation representation.

Consider now the example depicted in Fig. 2. Some event happens at time *t1* resulting in the generation of *eventnotice-t1* by some sensor. This EventNotice affects *attribute1* or *object1* by assigning it a value and certainty instantiated as *propertyvalue1*. At time *t2* a second event occurs generating *eventnotice-t2*, which in turn affects *attribute1*, in this case by assigning it a new value and certainty in the form of *propertyvalue2*. *Eventnotice-t2* also becomes associated with *propertyvalue1* as it effectively marks the end of *propertyvalue1's* period of being in effect. A similar process occurs with the onset of the third event at time *t3*.

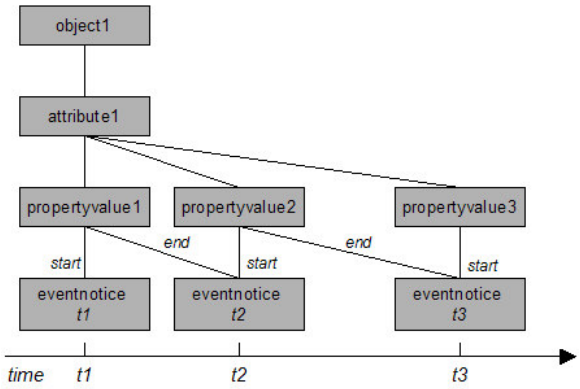

Fig. 2. *PropertyValues* delineated by *EventNotices*

The ontology permits a PropertyValue to be implemented as a DynamicSystem. What this means is the value and certainty functions are dynamically modeled and therefore they cause the PropertyValue to change even in the absence of new EventNotices. To illustrate the need for a DynamicSystem implementation of PropertyValues, consider the Position attribute of a PhysicalObject. The object's Position attribute's value at time t+1 is related to the object's Velocity (a vector providing speed and direction) at time t. Even if no new EventNotice affecting the position is received at time t+1, it is reasonable to assume that the object's position has changed. In the absence of additional information (*e.g.,* acceleration, trajectory) it might be reasonable to assume that the object continues to move with its last noted speed and direction until informed otherwise, albeit with increasing uncertainty as time goes on.

To be able to make such projections in the absence of explicit sensory information requires predictive models. It is for this reason that the SAW ontology shows DynamicSystems as a way of *implementing* PropertyValues. Certain attributes, such as Position, would be modeled by dynamic systems that might themselves generate internal EventNotices to update the attribute values, with some lesser degree of cer-

tainty, until new external sensory information arrives. It might also become desirable to fuse multiple model-predicted values or to combine model-generated values with sensory information in cases where the certainty of the external information is less than perfect.

2.2 Extending the SAW Core Ontology to a Specific Domain

The SAW Core ontology defines the fundamental classes and properties needed to support the representation of a large class of situations, specifically those that can be described by objects, attributes and relations that evolve over time. The core ontology is not very useful, however, until it is extended to a specific domain. There are two aspects to extending the SAW Core ontology to a given domain. The first involves sub-classing the *SituationObject* and *Attribute* classes, which define what kinds of objects are permitted as well as how they are to be described. The second involves sub-classing the *Relation* class to define the types of higher-order relations (i.e., Level 2 information) that are possibly relevant to the domain. These two processes are described in the following sub-sections. Two domain-specific sub-ontologies are used in these discussions; each is an example of a part of what might be a larger, all inclusive ontology for annotating military battlefield situations.

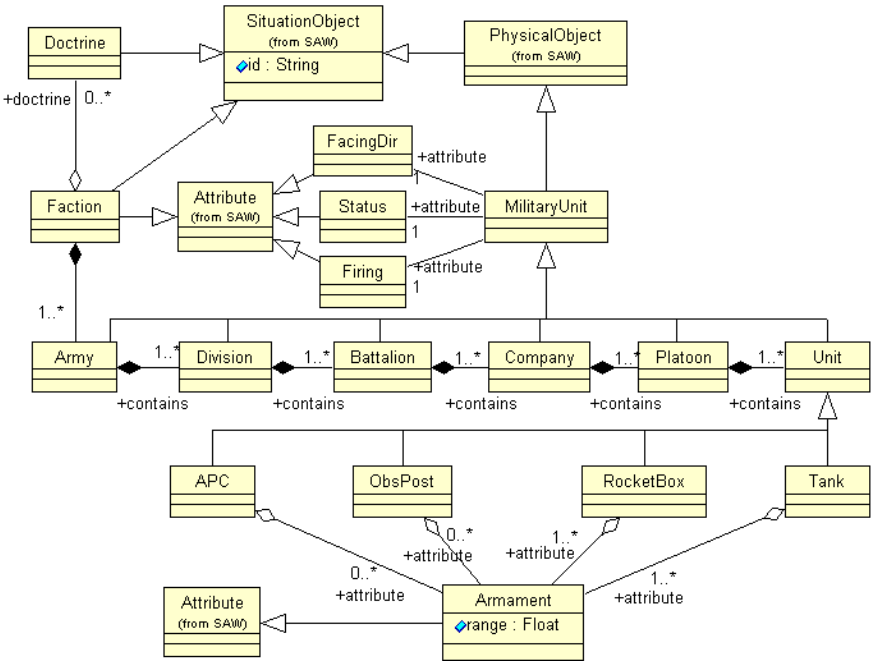

Fig. 3. Military Unit Ontology (partial realization)

Defining SituationObject and Attribute Sub-Classes. The natural place to begin extending the SAW Core ontology to a specific domain is with the definition of the SituationObjects pertinent to the domain. Fig. 3 shows a UML diagram of a Military Unit ontology intended to describe individual military units, their attributes and the ways they can be aggregated. Note that all of the defined classes are descendants of either the *SituationObject* class or the *Attribute* class defined in the SAW Core ontology, as indicated by the "(from SAW)" notes. The MilitaryUnit class is the super class for all other unit classes and it defines the Attributes shared by all of them: FacingDir, Status, and Firing. Note that this definition of MilitaryUnit is not meant to be complete but rather to illustrate the concept.

Defining Relation Sub-Classes. After defining the classes of objects pertinent to the domain comes the task of defining the relations among these objects that might be of interest. This process is accomplished by sub-classing the SAW Core ontology's *Relation* class. Fig. 4 depicts in UML a Battlefield Relation ontology that partially achieves this for the battlefield domain. While most of the relation sub-classes are partially defined, one, *FiringAt*, is fully defined by specifying its subject and object. Each relation sub-class must specify the classes of its operands (which may be more than two) in this way and the cardinality on these properties must be one; here, only *FiringAt* is shown completely so as to reduce the complexity of the diagram.

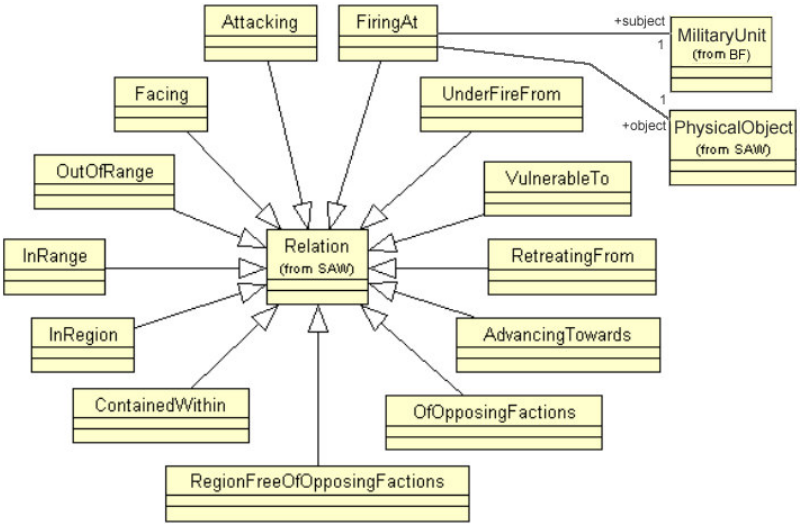

Fig. 4. Battlefield Relations

The reason the subject and object properties are needed has to do with the type of reasoning that we ultimately want to do concerning the relations occurring in a situa-

tion. This reasoning requires that we can identify the set of objects in the situation that can possibly be used in the satisfaction of a rule. By defining the permissible classes of operands for a relation we are effectively defining "sorts" which can be used to identify which objects the relations can be applied to. This becomes important when we develop rules for these relations that define when they hold true in a situation. Since RuleML, which we use to represent the rules for relations, does not have the capability of limiting the assignment of variables used in predicates to classes of objects, it is necessary for us to represent this information in the domain-specific relation ontologies.

The following OWL code fragment provides a concrete example of how these relation sub-classes are defined when we convert from UML to OWL; this fragment shows a portion of the code for the battlefield relation ontology highlighting the *FiringAt* relation, which we will be discussing further in the next section on RuleML:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY saw
     'http://www.vistology.com/onts/SAW/saw-core#'>
  <!ENTITY mu
     'http://www.vistology.com/onts/SAW/BF/units#'>]>
<rdf:RDF
  xmlns:owl ="http://www.w3.org/2002/07/owl"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:saw ="http://www.vistology.com/onts/SAW/saw-
core#"
  xmlns:mu=
         "http://www.vistology.com/onts/SAW/BF/units#">
 …
  <owl:Class rdf:ID="FiringAt">
   <rdfs:subClassOf resource="&saw;Relation"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#subject"/>
     <owl:allValuesFrom
          rdf:resource="&mu;MilitaryUnit"/>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#object"/>
     <owl:allValuesFrom
          rdf:resource="&saw;PhysicalObject"/>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>
 …
</rdf:RDF>
```

3 Domain Theory Rules and RuleML

For a system to be able to detect when a relation exists in a situation it needs to know the specific conditions that must be present for the relation to hold true. This knowledge is most readily represented using *implication*. With implication we can specify that Y (the consequence) can be taken to hold true when $X_1 \ldots X_n$ (the antecedents) are all true. One of the simplest approaches for representing such implication is through Horn clause statements [18], which is what we have selected for our system. Unfortunately, general Horn clause statements are not explicitly representable using the primitives in OWL. OWL can represent simple implication such as subsumption, but it has no mechanism for defining arbitrary, multi-element antecedents.

It is possible to construct an OWL ontology for representing Horn clause statements that could be used to implement rules as instance annotations, however, the rules would be rather verbose, extremely difficult to read and write, and would not be based on an accepted standard. An alternative approach is to use something outside of OWL. There are many languages one could choose for this purpose, but with the benefits afforded by XML (which our system exploits extensively) and the fact that OWL is XML-based, an XML-based language such as RuleML is really the only approach we would seriously consider. Since the DAML/OWL community decided last year to more closely align its rule effort with that of RuleML, RuleML currently appears as the most logical choice for representing rules in the context of OWL ontologies.

Our approach to the rule construction process is that of building RuleML rules on top of OWL ontologies. We assume that the ontologies are pre-defined and that all their classes are available to be used as elements in the rules. The question is which of the classes from the ontologies are to be used and in what way? Since the purpose of the rules is to permit definition of the conditions that must hold for relations to exist, the heads of the rules clearly must make reference to the domain-specific subclasses of the SAW Core class *Relation*. To do this we simply provide the URI of the desired relation class as the content of the <rel> tag in the head atom operator of a RuleML rule. For example:

```
<_head><atom><_opr><rel>bf:FiringAt</rel>…
```

The bodies of the rules contain additional atoms with <rel> tags that contain URIs pointing either to relation classes (just as is done in the head) or to domain-specific subclasses of the SAW Core class *Attribute*.

3.1 An Example of a Domain Theory in RuleML

A domain theory can be viewed as a network that defines the inter-relationships among the theory's axioms or rules. If we select a single node representing the head of a rule and consider only its descendants, we can create a view that forms a tree (with possibly replicated nodes). Such a tree representing a subset of a hypothetical domain theory for battlefield scenarios is depicted in Fig. 5. The nodes labeled with

predicates represent relations or attributes from the domain-specific ontologies. Relation predicates that have child nodes represent heads of rules and their children represent the antecedents or bodies for individual rules; these are denoted with solid dots and the links projecting from them are combined by arcs to indicate that they are to be taken conjunctively.

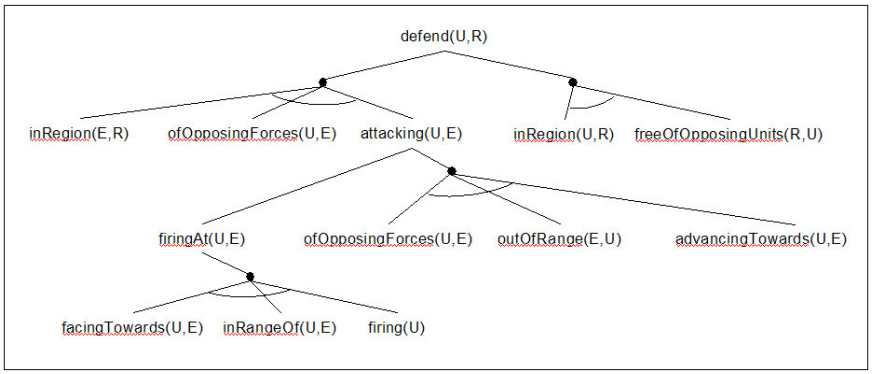

Fig. 5. Domain Theory Tree for defend(U,R)

The partial sub-theory shown in Fig. 5 (intended for illustrative purposes only) defines what it means for a unit U to ***defend*** a region R. The root of the tree is the relation *defend(U,R)* and its two children represent two different rule bodies, either of which, if satisfied, would imply the satisfaction of the root node relation. The first rule body contains three relations, the third one of which -- *attacking(U,E)* -- has two child nodes (i.e., rule bodies) of its own. Considering the first of these we see that one way to satisfy the relation *attacking(U,E)* is to satisfy the relation *firingAt(U,E)*. This relation in turn is satisfied by conjunctively satisfying the two relations *facing(U,E)* and *inRange(U,E)* as well as satisfying the attribute predicate, *firing(U)*. Note that in our approach, both relations and attributes are represented as predicates. The only difference between the two is that attribute predicates can only pertain to one *SituationObject* at a time whereas relation predicates can and usually do deal with multiple *SituationObjects*.

Here are RuleML rules for *attacking(U,E)* and *firingAt(U,E*:

```
<?xml version="1.0"?>
<!DOCTYPE rulebase SYSTEM
  "http://www.dfki.uni-kl.de/ruleml/dtd/0.8/ruleml-
datalog-monolith.dtd">

<rulebase
  xmlns:saw="http://vistology.com/onts/SAW/saw-core#"

xmlns:bf="http://vistology.com/onts/SAW/BF/battefield#">
…
```

```
<imp name="Attacking">
 <_head>
  <atom>
   <_opr>
    <rel>bf:Attacking</rel>
   </_opr>
   <var>X</var>
   <var>Y</var>
  </atom>
 </_head>
 <_body>
  <and>
   <atom>
    <_opr>
     <rel>bf:FiringAt</rel>
    </_opr>
    <var>X</var>
    <var>Y</var>
   </atom>
  </and>
 </_body>
</imp>

<imp name="Attacking">
 <_head>
  <atom>
   <_opr>
    <rel>bf:Attacking</rel>
   </_opr>
   <var>X</var>
   <var>Y</var>
  </atom>
 </_head>
 <_body>
  <and>
   <atom>
    <_opr>
     <rel>bf:OfOpposingFactions</rel>
    </_opr>
    <var>X</var>
    <var>Y</var>
   </atom>
   <atom>
    <_opr>
     <rel>bf:OutOfRange</rel>
    </_opr>
    <var>X</var>
    <var>Y</var>
   </atom>
```

```
   <atom>
    <_opr>
     <rel>bf:AdvancingTowards</rel>
    </_opr>
     <var>X</var>
     <var>Y</var>
   </atom>
  </and>
 </_body>
</imp>

<imp name="Firing At">
 <_head>
  <atom>
   <_opr>
    <rel>bf:firingAt</rel>
   </_opr>
   <var>X</var>
   <var>Y</var>
  </atom>
 </_head>
 <_body>
  <and>
   <atom>
    <_opr>
     <rel>bf:Facing</rel>
    </_opr>
    <var>X</var>
    <var>Y</var>
   </atom>
   <atom>
    <_opr>
     <rel>bf:InRangeOf</rel>
    </_opr>
    <var>X</var>
    <var>Y</var>
   </atom>
   <atom>
    <_opr>
     <rel>bf:Firing</rel>
    </_opr>
    <var>X</var>
   </atom>
  </and>
 </_body>
</imp>
…
</rulebase>
```

The URI references to relation and attribute sub-classes from the ontologies are underlined and in bold to highlight the location of their use. Note that there is no distinction made in the rules to indicate whether a predicate is a relation or an attribute. Our system actually needs this information for a process called *relevance determination* in which it derives relevant relations and attributes. Although the information is not in the rules it is in the ontologies and it is possible, using XSLT scripts, to determine the type of a predicate by looking up its URI in the ontology and climbing the inheritance hierarchy to determine whether the URI class is a descendant of *Relation* or *Attribute*.

4 Suggestions for RuleML

Our experience using RuleML in conjunction with OWL ontologies in the context of SAW reasoning has identified some possible additions to the evolving RuleML specification:

- We recommend the adding the option of being able to indicate the permitted classes of values that the variables in RuleML operators may take on. We understand that this may pose some issues for the underlying semantics of RuleML as it is defined currently. However, given the development of three versions of OWL– Full, DL and Lite – we believe a similar approach could be taken with RuleML that would permit applications to take advantage of the convenience of having sort-restricted variables at the possible expense of foregoing certain semantically desirable characteristics. Our work around for the absence of this capability in RuleML was to incorporate argument type information in the definitions of the relations in the domain specific ontologies. This approach has the disadvantage that both the rules and the ontologies need to be present in order to determine the permissible types for arguments in a rule.

- For our purposes it is necessary to be able to uniquely identify specific rules so we can pull a subset of relevant rules out of a larger domain theory (see [5,8]). As shown in Code Fragment 1, we have introduced a "name" attribute to the <imp> tag for this purpose. We considered using the rdf:ID attribute but it is more convenient for us to be able to have multiple rules with the same name, which rdf:ID does not permit. Having an rdf:ID attribute in addition to a name attribute would likely be useful for situations requiring uniquely identified rules even if they pertain to the same head.

- It seemed contrary to the spirit of XML to have to write the string representation of a URI as the text node of a <rel> tag. It would be more natural to be able to use an rdf:IDREF attribute on the <rel> tag, but the XML Schema for RuleML 0.8 (monolith version) [19] does not permit use of this attribute. Of all our recommendations accepting this one would seem to provide the most benefit as far as making it easier for RuleML to work more closely with OWL.

- The syntax of RuleML rules seems excessively verbose for writing basic rules. The authors have not been a part of the design discussions for RuleML and we accept that there are good reasons for the current syntax. Even so, it is difficult to see why a reduced syntax such as

```
<rule>
  <head>
    <rel/><var/>…
  </head>
  <body>
    <rel/><var/>…
    <rel/><var/>…
    …
  </body>
</rule>
```

  could not be part of some human-user friendly dialect of RuleML, similar in spirit to the proposed OWL Presentation Syntax [20].

5 Summary

We presented a case for using RuleML to construct domain theory rules on top of OWL ontologies. The context for this work is that of reasoning about situation awareness, for which we are currently developing a Situation Awareness Assistant (SAWA). We presented a SAW Core ontology and showed how it can be extended to handle domain-specific situations. Since our system needs to be able to derive higher-order relations that exist among the objects in a situation we need to be able to encode domain theories that specify the conditions under which specific relations come into being. This need requires the representation of general implication, a capability that goes beyond the simple implications available in OWL. We argued that RuleML not only provides general implication in the form of Horn clauses but also that its XML representation makes it the ideal chose for use with OWL. We then showed how we use RuleML to define rules that reference relation subclasses in our core and domain-specific ontologies. We concluded with four recommendations for additions to the RuleML specification: 1) permit constrains to be imposed on the classes of values that variables can assume, 2) add "name" and/or "rdf:ID" as possible attributes to the <imp> tag, 3) add "rdf:IDREF" as a possible attribute of the <rel> tag, and 4) create an alternative representation that is easier for humans to process.

Acknowledgements

This work was partially funded by the Air Force Research Laboratory, Rome, NY under contract numbers F30602-02-C-0039 and F30601-03-C-0076.

References

1. M. Endsley and D. Garland, *Situation Awareness, Analysis and Measurement,* Lawrence Erlbaum Associates, Publishers, Mahway, New Jersey, 2000.
2. J. Barwise, "Scenes and other situations", *J. Philosophy* 77**,** 369-397, 1981.
3. J. Barwise, *The Situation In Logic*, CSLI Lecture Notes 17, 1989.
4. A. Steinberg, C. Bowman, and F. White, Revisions to the JDL data fusion model, In Proceedings of SPIE Conf. Sensor Fusion: Architectures, Algorithms and Applications III, volume 3719, pages 430-441, April 1999.
5. C. Matheus, M. Kokar and K. Baclawski, Phase I Final Report: A Formal Framework for Situation Awareness. AFRL Funding Number: F30602-02-C-0039, January 2003.
6. K. Baclawski, M. Kokar, J. Letkowski, C. Matheus and M. Malczewski, Formalization of Situation Awareness, In Proceedings of the Eleventh OOPSLA Workshop on Behavioral Semantics, pp. 1-15, November, 2002.
7. C. Matheus, M. Kokar and K. Baclawski, A Core Ontology for Situation Awareness. Proceedings of FUSION'03, Cairns, Queensland, Australia, July 2003.
8. C. J. Matheus, K. Baclawski and M. M. Kokar, Derivation of ontological relations using formal methods in a situation awareness scenario, In Proceedings of SPIE Conference on Mulitsensor, Multisource Information Fusion, pages 298-309, April 2003.
9. Specware: Language manual, version 2.0.3. Technical report, Kestrel Institute, 1998.
10. SNARK: SRI's new automated reasoning kit, http://www.ai.sri.com/stickel/snark.html, 2002.
11. C. Matheus, M. Kokar and K. Baclawski, Phase II Proposal: A Formal Framework for Situation Awareness. AFRL Proposal Number: F2-1341, November 2002.
12. OWL Web Ontology Language XML Presentation Syntax. http://www.w3.org/TR/owl-xmlsyntax/.
13. The RuleML Initiative, http://www.ruleml.org/.
14. Resource Description Framework (RDF). http://www.w3.org/RDF/.
15. Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In Proc. of the 2003 International Semantic Web Conference (ISWC 2003), 2003.
16. OntoWeb Consortium, OntoWeb Deliverable 1.3: A survey on ontology tools, May 2002. http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OntoWeb_Del_1-3.pdf.
17. Kogut, P. A., Cranefield, S., Hart, L., Dutra, M., Baclawski, K. and Kokar, M. M. and Smith, J. E. UML for Ontology Development, *The Knowledge Engineering Review*, Vol. 17 (1), pp. 61 - 64, 2002.
18. A. Horn, "On sentences which are true of direct unions of algebras", *Journal of Symbolic Logic*, 16, 14-21, 1951.
19. RuleML DTDs, http://www.dfki.uni-kl.de/ruleml/indtd0.8.html.