Position Paper: Using Ontology-based Rules for Situation Awareness and Information Fusion

Christopher J. Matheus
Versatile Information Systems, Inc.
5 Mountainview Drive
Framingham MA 01701 USA
cmatheus@vistology.com

Background

Versatile Information Systems, Inc. (VIS) has been applying Semantic Web technologies to the problems of situation awareness and information fusion for more than five years. **Situation awareness** involves the real-time processing of event-based information coming from an evolving situation in an attempt to understand what is happening. In our view, situation awareness primarily comes down to identifying higher-order relations that come into being within a situation and that have particular relevance to the problem at hand as defined by the user's goals or objectives[1],[2]. By higher-order relations we mean relations involving multiple objects; OWL ObjectProperties represent the simplest of such relations involving two objects but situation awareness is often interested in more complex relations involving several objects. We contrast these higher-order relations with those that merely define characteristics of an individual object; DataProperties fall into this category. The analytical processes that go into establishing situation awareness necessarily involve **information fusion**, the processes by which data/information from multiple sources are combined to produce new enhanced information that incorporates aspects of the raw, original sources.

Our approach to both situation awareness and data fusion involves the use of **formal ontologies** to describe the fundamental events, objects and relations of a situation's domain and **logical rules** to define ways of fusing information and identifying higher-order relations relevant to the situation at hand. We have been working with **DAML/OWL** for representing our formal ontologies since the language's inception. For rules we began by using **RuleML**, which we systematically converted to Jess[3] rules using XSLT scripts. The rule we have written have always been grounded in the classes and properties of one or more OWL ontologies; we refer to this as constructing rule-based domain knowledge *on top of* OWL ontologies[4]. With the advent of SWRL in the fall of 2003 we adopted it as our primary rule language because of its close coupling with OWL, our primary language of discourse. We have since developed a graphical SWRL rule editor called RuleVISor, which we are planning to release to the community in the near future. A distinguishing feature of RuleVISor is that it greatly facilitates the development of rules based on ontologies by permitting the user to drag and drop OWL ontology elements into the head and body of a rule.

Challenges in Using SWRL on top of OWL

We strongly advocate the use of formal methods for representing ontologies and reasoning with rules. We have attempted to whole-heartedly adopt OWL and SWRL as formal languages for the use in our research and development systems. Doing so however has come at a cost due to several challenges encountered in trying to use these languages for practical applications.

Struggles with the Binary-Property Limitation

The restriction to binary properties and the lack of a join mechanism in OWL has forced us at times to construct rather unnatural ontologies and/or represent complex properties in the form of rules rather than more simple ontological constructs. In SWRL, the restriction to the use of binary predicates that it inherited from OWL has significantly hampered our ability to readily codify domain knowledge. We find ourselves first writing rules using higher-order predicates and then manually translating them into SWRL rules with the addition of "relation classes" to capture the inherently higher-order nature of the rules we wish to construct. For example to represent the head of a rule about a person being the uncle of another person at a particular time – uncle(?X,?Y,?T) – we might construct an instance of a relation class called "UncleAt" and give it three properties, one for each of the variables in the original ternary predicate, as such:

```
  <ruleml:head rdf:parseType="Collection">
      <swrlx:classAtom>
        <owlx:Class owlx:name='&vis;UncleAt'/>
          <ruleml:var>?X</ruleml:var>
        </swrlx:classAtom>
      <swrlx:datavaluedPropertyAtom
           swrlx:property='&vis;firstPerson'>
          <ruleml:var>?X</ruleml:var>
          <ruleml:var>?P1</ruleml:var>
      </swrlx:datavaluedPropertyAtom>
      <swrlx:datavaluedPropertyAtom
           swrlx:property='&vis;secondPerson'>
          <ruleml:var>?X</ruleml:var>
          <ruleml:var>?P2</ruleml:var>
      </swrlx:datavaluedPropertyAtom>
      <swrlx:datavaluedPropertyAtom
           swrlx:property='&vis;time'>
          <ruleml:var>?X</ruleml:var>
          <ruleml:var>?T</ruleml:var>
      </swrlx:datavaluedPropertyAtom>
    </ruleml:head>
```

This approach forces us to violate the SWRL prohibition of using unbound variables in the heads of rules (i.e., the "safety" condition) because we need a way to "generate" the relation instance. What we are trying to say in the rule is "when the antecedent of the rule is met, an instance of the relation class uncleAt should be added to the fact base". This construction is easy to implement in our underlying inference engine which is happy to perform a gensym to generate a new identifier for the instance and then add the appropriate triples required to capture the values of the instance's properties. Here is the Jess code to do this for the uncleAt rule as generated by our RuleVISor editor:

```
(defrule "Uncle At"
  (triple (p "vis:brotherOf") (s ?P1) (o ?P3))
  (triple (p "vis:parentOf") (s ?P3) (o ?P2))
  (triple (p "vis:global") (s "vis:System") (o ?T))
  =>
  (bind ?X (gensym*))
  (assert (triple (p "rdf:type") (s ?X) (o "vis:UncleAt")))
  (assert (triple (p "vis:firstPerson") (s ?X) (o ?P1)))
  (assert (triple (p "vis:secondPerson") (s ?X) (o ?P2)))
  (assert  (triple (p "vis:time") (s ?X) (o ?T)) )
)
```

According to the SWRL draft definition[5], the safety condition (which we are violating here) need not ever be violated "because existentials can already be captured using OWL someValuesFrom restrictions". I am unable to see how someValuesFrom would resolve the problem in this case and in fact this issue was raised last fall on the www-rdf-logic mailing list by Enrico Franconi and no one seems to have a satisfactory answer to the problem[6].

The Pains of Built-ins

We have implement a large number of the SWRL built-ins in our enhanced Jess reasoning engine, focusing mostly on those that we needed for our own purposes and those for which we had pre-existing code. Apart from the fact that there are just way too many built-ins to have to write code for, there is a big challenge presented by the fact that the built-ins are defined as relations with no explicit input/output designations assigned to their arguments. In our rules we have always found that we need to use the built-in capabilities as if they were functions in which you specify the values for all but one of the arguments, which becomes your output variable. What this lack of input/output designation in SWRL built-ins means from an implementation perspective is that the code for the built-ins must determine which of the arguments is supposed to be the output variable and then select the appropriate functional method to apply to the other arguments.

For example, consider `swrlb:add` which can be applied to an arbitrary number of two or more arguments with the first argument being the sum of the remaining arguments. If you use the atom (swrlb:add ?X 1 2) the processor of this statement must detect that the first argument is unbound (i.e. a variable) and thus it becomes the output argument. Knowing that the value of the first argument is to be calculated the processor must apply its *summation* method to the remaining arguments. If, on the other hand, you pass (swrlb: 10 ?X 5) to the processor it needs to figure out that it should *subtract* 5 from 10 to calculate the value of the variable ?X. In this case we have used swrlb:add to do *subtraction* even though there is also a swrlb:subtract builtin. Now consider the even more bewildering case where more than one argument is unbound: (swrlb:add ?X 100 ?Y). What should the processor do in this case? According to the semantics of SWRL this is perfectly legal but it is not at all clear how, in a practical system, one would return the infinite set of tuples that the result is composed of. In our system we treat this case an error, which for all conceivable kinds of applications our group is interested in it would be.

The Ugliness of Time and Uncertainty

In a situation, things happen and things change. When writing rules it is generally easier to simply assume that everything in the rule is referring to the most current moment and ignore time all together. This approach actually works quite well when dealing with an object-oriented system where each property of a specific object maintains its most recent value; in other words when a new value for a property comes it, it overwrites the previous value. In this case, rules that we might call “time agnostic” work well. If, however, you want to play “by the rules” in the OWL/RDF world you cannot use this trick because of RDF’s monotonicity assumption – once a triple is asserted it can not be retracted or changed.

You can get around this problem in a couple of ways. In the first approach you treat new information outside of the OWL/RDF context. When new information comes in you check to see if it would introduce an inconsistency in your current triples base representing the situation at hand and if it does not you add it, otherwise you attempt to reconcile the inconsistency from outside box. Doing this reconciliation may involve the need to retract triples that you decide no longer hold to be true – this is non-monotonic but we are doing it outside the OWL/RDF context in order to make a new triples base that can me treated monotonically. Unfortunately the retraction of triples is not so easy to do because there may be other triples that were inferred fby the axioms of OWL/RDF/XSD from the triples you wish to remove. We initially tried this approach and found that the logical retraction capability in Jess was not able to do the recursive retraction in all cases and we suspect there may be some pathological cases involving cycles in which it may be simply impossible.

The second approach is to timestamp every value coming into the system. When you do this you can no longer ask for something as simple as ‘the position of a vehicle”. Instead you need to ask for the value at a specific time. Usually you want the most recent value but this requires a search through all historical values to find the one that is most recent (try to write a set of rules to do that efficiently!). Things become even more challenging when you know that the value of a property is constantly changing (e.g., the position of a moving vehicle) but the most recent value comes for a time in the past. What do you do when you need to use this property's value in a rule? Use the old value knowing fully well that this value is wrong? Write rules to try to predict the current value based on some model of how the property might be evolving? Use the most recent value but assign to it some lower level of "certainty'? All of these approaches present their own challenges and further support the claim that using rules to reason about time variant values in a monotonic context is hard. Using SWRL and trying to write rules based on binary predicates only makes things that much harder.

Did someone mention "uncertainty'"? It is often the case in situation awareness that the values coming in from the field (e.g., sensor readings, observation reports) carry with them a level of certainty (or conversely, uncertainty), usually represented by some numeric measure or measures. This certainty measure comes into play in rules in a manner similar to how the timestamp we talked about adding to every value complicated matters. Every value must now have a certainty measure (even if it is completely certain) and these values need to be propagated through the firing of rules; in other words when the firing of a rule results in the insertion of a new value for some (possibly complex) property, its certainty level needs to be calculated from the certainty levels of the values used in the antecedent of the rule. Again, as writers of rules we really do not want to have to always worry about including certainty with every property value used, or with specifying how those uncertainties should be combined. And we don't even want to think about what happens when we want to know the certainty of a value at the current moment but the certain level in the fact base is from some time in the past… Can you image what a simple SWRL rule would look like if it accounted for both time varying values and certainties for those values? We are exploring some ideas in this area but they will almost certainly involve solutions outside the context of rules alone.

Rule Use Cases

Since one of the objectives of this workshop is to identify use cases for rule languages here are some of the rule-based research and development applications we have completed or are currently involved with:

- ConsVISor is VIS' heuristic-based consistency checker for OWL/RDF documents developed as part of the DAML Project. Its heuristics are grounded in the axioms and semantics of XSD, RDF, RDFS, and OWL (all flavors) and are implement as sets of **Jess rules**. ConsVISor is available as a free Web service at http://www.vistology.com/consvisor.
- SAWA is a Situation Awareness Assistant being developed under a Phase II SBIR grant from the Air Force Research Laboratory, Rome NY[7]. The domain knowledge for this effort is represented in OWL and **SWRL**. The specific domain we have focused on to date is that of Supply Logistics. Portions of this research were demonstrated at ISWC 2004[8]. A byproduct of this work is our graphical SWRL rule editor, RuleVISor, which we soon plan to make available for free via download off of our corporate Web site at http://www.vistology.com/rulevisor.
- SIXA is a Semantic Information Exchange Architecture being developed under an STTR grant from the Office of Naval Research. Its primary purpose is to provide the means for the mediation of data/information through Semantic Information Services that negotiate the interchange of information. One aspect of this work is the development of a mechanism for reasoning about data pedigree, represented using an OWL-based pedigree ontology, using **SWRL** rules to assist in the selection of the most reliable sensor data concerning the tracks of naval vessels.
- A project dealing with intelligence information analysis for the Army is in its early stages. While we are not able to discuss it in detail we can state that we are planning on using OWL and **SWRL or RuleML** to capture domain knowledge about evolving enemy courses of action (ECOA). This knowledge will involve complex time-based relationships

among a potentially large number of entities. The representation of time and uncertainty and the ability to predict activities that might occur in the future will be important capabilities.

Position Summary

With our focus on real time situation awareness and information fusion we are regularly pushing the practical limits of Semantic Web technologies. We have encountered several significant challenges in trying to develop systems that involved SWRL rules defined over OWL ontologies;

- The restriction to binary predicates in both SWRL and OWL – leads to convoluted rules when dealing with complex relations.
- The safety condition in SWRL – makes it difficult if not impossible to write rules that result in the assertion of complex relations.
- Implementing within a reasoning system the SWRL built-ins as they are currently defined – being defined as relations rather than functions complicates the codification.
- Dealing with time under monotonicity constraints and in general – many issues here with no clear near-term remedy.
- Dealing with uncertainty – ditto, but this one is not likely to be a priority issue for this group.

Given the difficulty involved in using SWRL for real-world applications it seems likely that we will move back to RuleML for future applications unless some of the inherent limitations of SWRL are remedied. This shift will not resolve all of our concerns but it will make rule development much less painful. The one most significant change that could be made to SWRL that would likely change our opinion of its usefulness would be coming up with a way to permit the use of higher-arity structures as atoms within the heads and bodies of rules. The restriction to binary properties is understandable within OWL given the desire for tractability and decidability. It seems that when we move into the realm of rules where we must discard these qualities anyways that we should no longer be required to be crippled by the binary-property restriction of OWL/RDF. Unfortunately it is not immediately clear how this might be done.

References

[1] C. Matheus, K. Baclawski and M. Kokar, Derivation of ontological relations using formal methods in a situation awareness scenario. In Proc of SPIE Conference on Multisensor, Multisource Information Fusion, pages 298-309, April 2003.

[2] 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.

[3] Jess: the Rule Engine for the Java Platform. http://herzberg.ca.sandia.gov/jess/.

[4] C. Matheus, K. Baclawski, M. Kokar, and J. Letkowski, Constructing RuleML-Based Domain Theories on top of OWL Ontologies. In Proceedings of Rules and Rule Markup Languages for Sematic Web: Second International Workshop, RuleML 2003, Sanibel Island, Florida, October 2003.

[5] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, April 2004. http://www.daml.org/rules/proposal/.

[6] http://lists.w3.org/Archives/Public/www-rdf-logic/2004Nov/0017.html

[7] C. Matheus, M. Kokar, K. Baclawski, J. Letkowski, C. Call, M. Hinman, J. Salerno and D. Boulware. SAWA: An Assistant for Higher-Level Fusion and Situation Awareness. In Proc. SPIE Conference on Multisensor, Multisource Information Fusion, Orlando, Florida, March 2005.

[8] C. Matheus, M. Kokar, K. Baclawski. J. Letkowski and C. Call. The Practical Application of Semantic Web Technologies for Situation Awareness. Demonstration Paper, International Semantic Web Conference, Hiroshima, Japan, November, 2004. http://iswc2004.semanticweb.org/demos/11/paper.pdf.