An Application of Semantic Web Technologies to Situation Awareness

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

[1] Versatile Information Systems, Inc. Framingham, Massachusetts USA
cmatheus@vistology.com
http://www.vistology.com
[2] Northeastern University Boston, Massachusetts USA
ken@baclawski.com mkokar@ece.neu.edu
[3] Western New England College, Springfield, MA, USA
jletkows@wnec.edu

Abstract. Situation awareness involves the identification of relationships among objects participating in an evolving situation. This problem in general is intractable and thus requires additional constraints and guidance defined by the user if there is to be any hope of creating practical situation awareness systems. This paper describes a Situation Awareness Assistant (SAWA) based on Semantic Web technologies that facilitates the development of user-defined domain knowledge in the form of formal ontologies and rule sets and then permits the application of the domain knowledge to the monitoring of relevant relations as they occur in a situations. SAWA includes tools for developing ontologies in OWL and rules in SWRL and provides runtime components for collecting event data, storing and querying the data, monitoring relevant relations and viewing the results through a graphical user interface. An application of SAWA to a scenario from the domain of supply logistics is presented along with a discussion of the challenges encountered in using SWRL for this task.

1 Introduction

The essence of situation awareness lies in the monitoring of various entities and the relations that occur among them. Since the properties of relations, unlike the properties of objects, are not directly measurable, one needs to have some background knowledge (such as ontologies and rules) to specify how to derive the existence and meaning of particular relations. For instance, in the domain of supply logistics, relations like "suppliable" or "projected undersupply within 2 days" need to be systematically specified. The number of potentially relevant relation types is practically unlimited. This presents a great challenge to developers of general-purpose situation awareness systems since it essentially means that such systems must have the potential to track any possible relation. In other words, the relation determination algorithms must be generic, rather than handcrafted for each special kind of relation. Furthermore, in order to derive a specific relation one often needs to access a number of data sources and then combine (i.e., fuse) their inputs. One way to address these challenges is to use generic reasoning tools, such as those based on the principles being employed by the Semantic Web. To take advantage of this approach, however, all information must be available in a formally defined knowledge base.

At Versatile Information Systems, Inc., we are developing a collection of flexible ontology-based information fusion tools needed for identifying and tracking user-defined relations. These tools collectively make up our **S**ituation **Aw**areness **A**ssistant (**SAWA**). The purpose of SAWA is to permit the offline development of problem specific domain knowledge and then apply it at runtime to the fusion and analysis of object-level data. Domain knowledge is captured in SAWA using OWL ontologies for describing the classes and properties of the domain and SWRL rules for defining the conditions of higher-order relations. The user controls the system situation monitoring requirements by specifying "standing relations", i.e., high-level relations or queries that the system is to monitor. SAWA provides a flexible query and monitoring language based on OWL-QL that can be used to request information about the current situation or to conduct what-if queries about possible future situations. In this paper we describe the structure and capabilities of SAWA and show its use on examples from the supply logistics domain. In particular, we show how to develop an appropriate ontology and associated rules, how SAWA collects and processes incoming events and how it communicates with the user. We also discuss the advantages and limitations of applying Semantic Web technologies to the problem of situation awareness.

2 General Approach

We view situation awareness as a fusion problem involving the identification and monitoring of higher-order relations among object-level objects. As mentioned in the introduction, practical solutions to this problem require user-defined constraints, which we usually identified with a corpus of knowledge specific to a domain of interest, otherwise known as *domain knowledge*. The use of domain knowledge requires a form of representation and a means for processing or reasoning about the knowledge representations. Rather than developing ad hoc representations we advocate the leveraging of existing standards. We also believe strongly in the value of formal representations that can be used in conjunction with generic yet formal reasoning systems. Our approach to domain knowledge representation, which we will describe shortly, is thus premised on use of standards-based formal representations.

Even with appropriate domain knowledge the number of possible relations definable within the domain knowledge constraints can remain intractable. To further constrain a situation we believe it is necessary to know something about the user's specific *goals*. By knowing more specifically what the user is looking for, automated systems can focus attention on just those events and candidate relations that are relevant. Our process for *relevance reasoning* has been reported elsewhere [1] and will not be explained in detail in this paper. We will summarize, however, by saying that relevance reasoning takes a *standing relation* (i.e. a goal) from the user, identifies the portion of the domain knowledge that is relevant to the standing relation, finds the attributes in the domain knowledge that must be grounded in input events and uses these attributes to identify what types of objects and which of their attributes need to be monitored in the event stream. With this mechanism, the large number of objects and attributes in a situation can be pared down to a more manageable stream of data in which only a comparatively small number of relevant relations must be monitored.

2.1 Ontology Representation in OWL

In our current efforts we have been exploring the use of recent developments for the Semantic Web [2]. In particular we have chosen to use the OWL Web Ontology Language [3] for defining ontologies that serve as the basis for data and knowledge representation within our situation awareness systems. The advantages of using OWL includes the fact that it is defined by a formal set of semantics and that there are a growing number of automated systems to formally process OWL documents, including editors, consistency checkers and reasoning engines [4].

OWL was designed to capture the classes, properties and restrictions pertinent to a specific domain. As such, OWL can capture basic class hierarchies, properties among classes and data and simple constraints on those properties and classes. OWL, however, cannot capture all types of knowledge relevant to a given domain. In particular, it does not provide a way to represent arbitrarily complex implications, in which knowledge of the existence of a collection of facts (X_1, X_2...X_n) implies the truth of some other information (i.e., $X_1 \Lambda X_2 \Lambda ... X_n \rightarrow Y$). For example, there is no way in OWL to define the relationship of "uncle(X,Y)" which requires knowing that X is male, X has a sibling Z, and Z has a child Y. The joining of collections of interrelated facts into implication rules as illustrated in this example is very common when defining relationships important to domains involving situation awareness. We therefore need the ability to define portions of our domain knowledge using a rule language, and for this purpose we have selected the Semantic Web Rule Language, SWRL [5].

2.2 Rule Representation in SWRL

SWRL is built on top of OWL and, like OWL, has a formally defined semantics, making it a natural choice for use in our situation awareness applications. SWRL does, however, have some shortcomings that make it less than ideal. Because it was officially introduced as a draft recommendation in just the spring of 2004, it is relatively new and is still evolving; this means there are few tools and applications for use with SWRL and it remains a moving target which may undergo radical changes that will introduce inconsistencies for early adopters. Furthermore, SWRL requires the use of binary predicates. While it is possible to represent concepts dependent on higher-arity relations using SWRL, the process of doing so significantly complicates the resulting rules, making them difficult to read and maintain. As an example consider the concept of a "part" at a "facility" being in "critical supply" at a particular "time", meaning there is a greater need for the part than the number of units available. What we would like to do is create a rule with a predicate of the form criticalPartAtFacility(?Part,?Facility,?Time,?DeficitAmount)[1] as its head and additional predicates in the body that define the conditions under which this predicate should be deemed to be true. To do this in SWRL we need to convert this four-term predicate into an instance of a class (fabricated solely for this rule) that is the domain of four properties, one for each of the four terms.

In the nine rules for our Repairable Assets scenario in which we monitor for critical and marginal parts at a number of airbases this technique was employed nine

[1] Variables in the examples of SWRL code presented in this paper are indicated by being prefaced with a question mark, such as in `?Facility`.

times and was usually repeated in the head and bodies of multiple rules. The need for this technique contributed greatly to the more than 1000 lines of SWRL code required to implement these nine relatively simple rules; this in turn made the debugging of the code very tedeous. Still, the advantages of SWRL – primary its formal semantics and its close association with OWL – were enough to encourage us to continue with our exploration of its use for situation awareness.

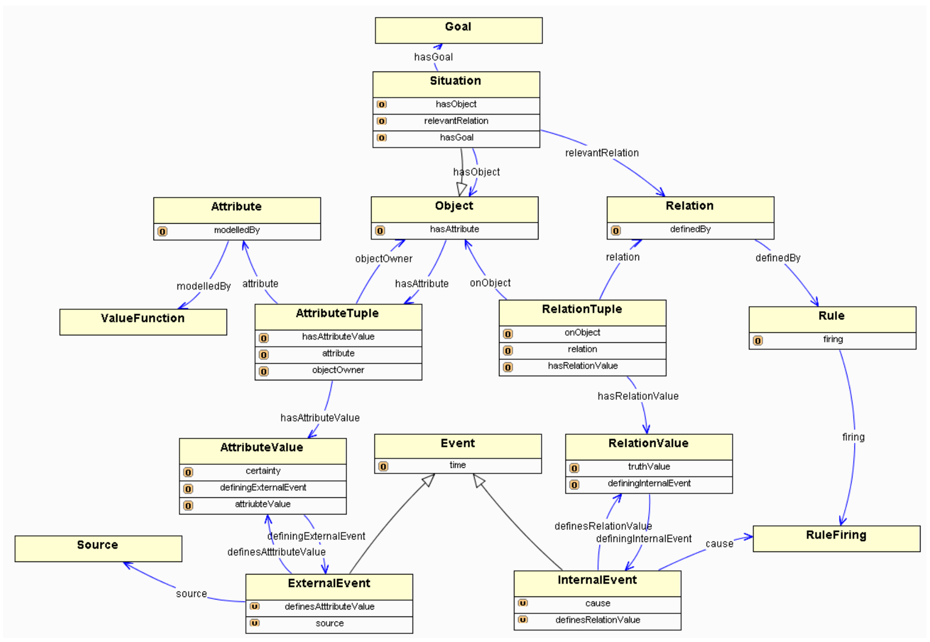


Fig. 1. SAW Core Ontology. This ontology serves as the basis for all domain specific ontologies and rule sets. According to the ontology a Situation consists of Objects and Relations and a Goal (standing relation). Objects have AttributeTuples that are associated with specific Attributes and a collection of AttributeValues defined according to ExternalEvents. Relations are realized through RelationTuples that connect pairs of Objects with RelationValues defining by the firing of Rules.

2.3 SAW Core Ontology

We are interested in building systems for situation awareness that are generic in nature. That is to say that the systems should be applicable to a wide variety of problem domains simply through the redefinition of the domain knowledge that they use. For this approach to work, some core concepts need to be established that will be used as the basis for the development of specific domain knowledge ontologies and rule sets. For this reason we have developed a SAW Core Ontology that serves as the representational foundation of all domain knowledge that is built on top of it. We have reported on this core ontology in earlier papers [6] and will not describe it in detail here. A simplified version of the ontology is shown in Fig. 1 with the key concepts being use of objects that have attributes with specific values being defined by external events that occur over time; in addition, relations combine pairs of objects with truth values defined over time by the firing of rules that define the relations.

3 SAWA High-Level Architecture

The SAWA High-Level Architecture has two aspects as shown in Fig. 2: a set of offline tools for Knowledge Management and a Runtime System of components for applying the domain knowledge to the monitoring of evolving situations. The knowledge management tools include an ontology editor, an ontology consistency checker and a rule editor. The runtime system consists of a Situation Management Component (SMC), an Event Management Component (EMC), a Relation Monitor Agent (RMA), a Triples DataBase (TDB) and a Graphical User Interface (GUI).

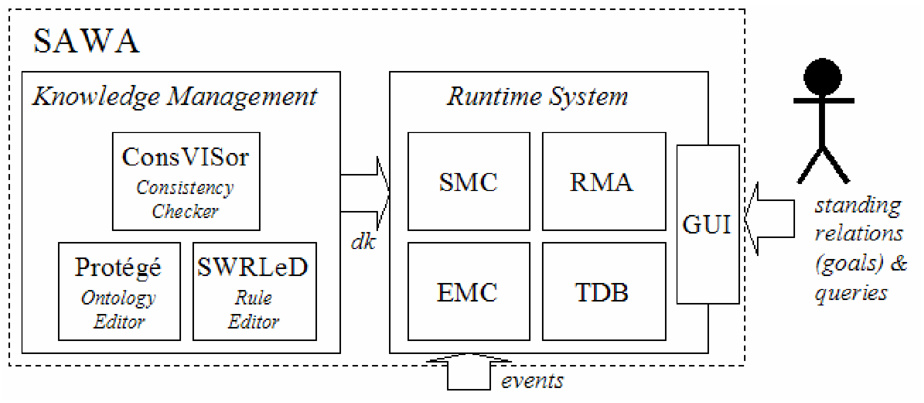


Fig. 2. SAWA High-Level Architecture. On the left side of the diagram is the Knowledge Management suite of tools used to develop the domain knowledge that serves as input to the Runtime System, shown on the right hand side. The user interacts with the system through the GUI by issuing standing relations (goals) and queries. Events from the outside world come into the runtime system and are processed for redistribution to other components by the Event Management Component (EMC).

4 SAWA Knowledge Management

Knowledge Management in SAWA is handled by a loosely coupled suite of tools for developing and maintaining OWL ontologies and SWRL rule sets.

4.1 Ontology Editor

The OWL language is based in RDF [7], which has an XML-based representation. As such, any text or XML editor could be used to develop OWL ontologies. The manual coding of OWL is, however, tedious and prone to error, making specialized editors highly desirable. There are a number of editors available for OWL [8] but the most widely used is Protégé [9]. Protégé is a general-purpose ontology management system developed long before OWL but for which OWL plug-ins have been developed. Using Protégé with the basic OWL plug-in permits the use of Protégé's frame-based editor to construct OWL classes, properties and restrictions among them as well as to develop annotations for OWL ontologies. This approach is adequate but not as convenient as a graphical editor that allows the visual display and manipulation of the relations between objects and properties. Fortunately there is a plug-in for Protégé called ezOWL that provides a graphical editor on top of the basic OWL-plugin. All of the ontologies depicted in this paper are screenshots taken from ezOWL. EzOWL has its limitations (for example it does not cleanly display more than two properties between two classes) and does not always produce correct OWL code, but it is currently the best available visual editor for OWL and does a satisfactory job, provided the resulting code is checked for consistency.

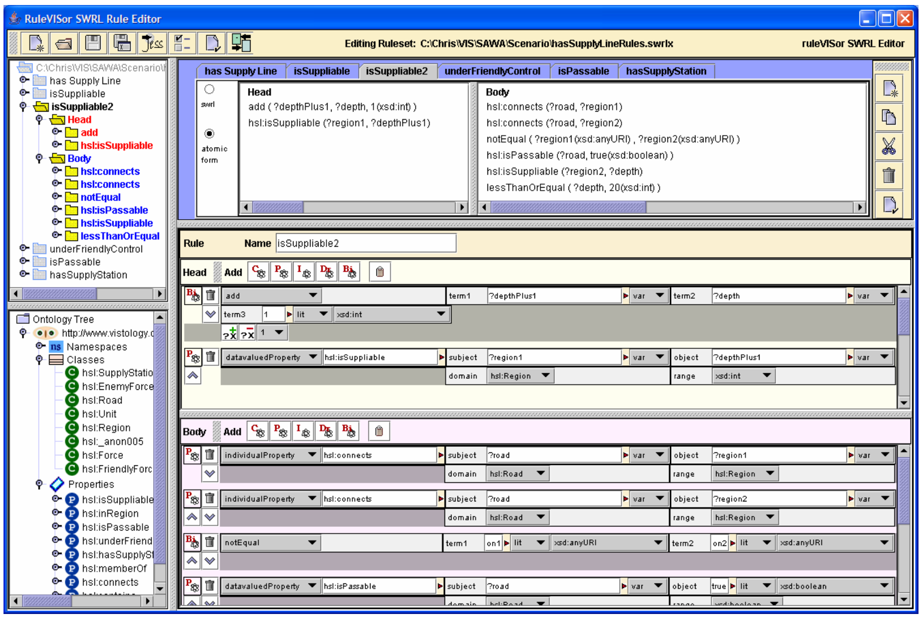
4.2 Consistency Checker

Developing an accurate and consistent ontology is not easy, particularly as the complexity of the domain increases. For all but the most trivial problems it is imperative that newly constructed ontologies be automatically validated for logical consistency; this is also invaluable when combining multiple ontologies that may individually be consistent but are collectively incompatible. It has been the authors' experience that seldom is the first design of an ontology complete and consistent, and the use of consistency checking tools has saved tremendous amounts of development time. SAWA includes ConsVISor [10], an OWL/RDF consistency checker, in its suite of knowledge management tools. ConsVISor is both a standalone Java application and a free Web Service available. at http://www.vistology.com/consvisor.

ConsVISor's purpose is to analyze OWL and RDF documents looking for symptoms of semantic inconsistencies. Not only does it detect outright semantic violations, it also identifies situations where logical implications have not been fully specified in a document. For example, if an ontology places a minimum cardinality constraint on a property for a specific class and an instance of that class is created without having the minimum number of property values, an informative message is provided as shown. Emphasis is placed on providing highly informative feedback about detected symptoms so as to aid the correction of underlying errors by the human user. ConsVISor's output however is based on an OWL-based Symptom Ontology [11] and as such can produce symptom reports in OWL that can be automatically processed by other OWL-cognizant programs.

4.3 Rule Editor

SWRL rules in their XML representation are syntactically and (frequently) semantically difficult to read and write. It was therefore decided that SAWA needed an easy to use editor to assist in the construction and maintenance of SWRL rules. With SWRL being so new, there were no SWRL editors available and so we decided to



Editing Ruleset: C:\Chris\VIS\SAWA\Scenario\hasSupplyLineRules.swrlx

Head
add (?depthPlus1, ?depth, 1(xsd:int))
hsl:isSuppliable (?region1, ?depthPlus1)

Body
hsl:connects (?road, ?region1)
hsl:connects (?road, ?region2)
notEqual (?region1(xsd:anyURI) , ?region2(xsd:anyURI))
hsl:isPassable (?road, true(xsd:boolean))
hsl:isSuppliable (?region2, ?depth)
lessThanOrEqual (?depth, 20(xsd:int))

Rule Name isSuppliable2

Fig. 3. RuleVISor. This screenshot of the RuleVISor SWRL editor shows its use on a set of rules used by the Supply Logistics scenario described in Section 6.

implement one, which we are calling RuleVISor. A screenshot of RuleVISor being used on a rule set for the Supply Logistics scenario described in Section 6 is shown in Fig. 3. The rules are displayed along the top left hand side of the editor in a directory style layout for easy selection and high-level scanning. The rule that is currently being edited appears in two forms in the right-hand section of the editor. At the top of this section is the display of the contents of the rule head and body in either an easy to read atomic form, which is shown in the screenshot, or as raw SWRL code (not shown). Below this display is the section where editing of the rule takes place, including the optional naming of each rule. This section is split into a portion at the top for editing the head followed by a portion for editing the body. Within either of these the user has the option of adding or deleting binary atoms, atomic atoms, instances, data value ranges and built-in functions simply by clicking on the appropriate icons. Each clause in a rule head or body appears in a three row region that provides the name of the atom, the terms it operates over and possibly other constraints such as term type restrictions. The values of the terms can either be typed in by the user or dragged from other areas of the editor. The primary source for dragged items is the Ontology Tree that appears in the lower left hand corner.

The Ontology Tree displays the contents of the ontologies upon which a rule set is to be built. Of most interest here are the Classes and Properties of the ontology, which are used to populate the term slots of atoms used in the rule heads and bodies. Class and Property names may be dragged to any text entry box in the editor but they will only be accepted by the box if the value being dragged matches the type that the box expects. This form of primitive type checking represents the beginning of a much more sophisticated policy for consistency checking based on ConsVISor that is planned for a future version of RuleVISor.

5 SAWA Runtime System

The SAWA Runtime System, also called the SAWA Engine, is depicted in Fig. 4 along with the communication channels between its sub-components. SAWA is implemented in Java, includes Jess as the basis for its reasoning functions and uses our proprietary RDF/OWL/XSD parser. The SAWA Engine consists of the following sub-components: the Situation Management Component (SMC) which is the system's central controller, the Event Management Component (EMC) which processes all incoming events, the Relation Monitoring Agent (RMA) which monitors relevant events for the status of relations occurring in the evolving situation, the Triples DataBase (TDB) which maintains a historical record of all situation events and permits the processing of queries, and the Graphical User Interface (GUI) which handles all user interaction with the system. The function of each of these components is described further in the subsections that follow.

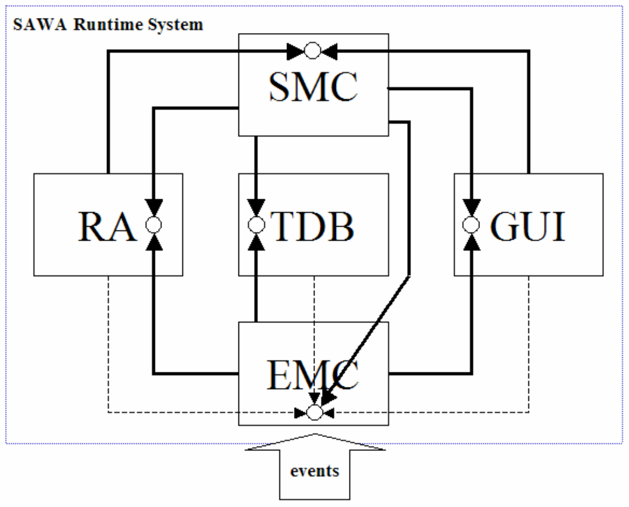


Fig. 4. SAWA Runtime System

5.1 Situation Management Component

The Situation Management Component (SMC) is the central controller for SAWA. It interacts with the GUI to provide options to the user and to accept the user's commands to start, stop and query situations. In addition, it serves as the communication channel between the GUI and the TDB and RMA. The SMC initializes the monitoring of situations by instructing the EMC to start listening to specific event streams and informs the RMA, TDB and GUI how to connect to the EMC to receive their appropriate streams of processed events. The SMC is also responsible for performing relevance reasoning, which is achieved through the application of XSLT scripts, and for passing the appropriate set of relevant rules to the RMA and the set of relevant objects and attributes to the EMC.

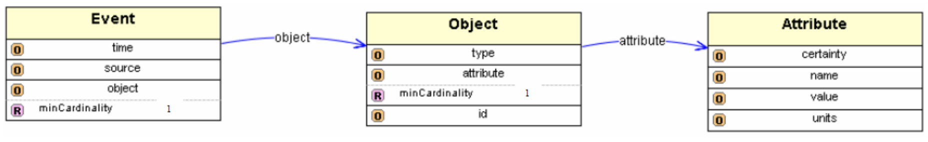


Fig. 5. Event Ontology. Simple ontology used to represent incoming events for processing by the EMC. Each Event describes one or more Objects each having one or more Attributes for which a value and certainty measure are defined.

5.2 Event Management Component

The Event Management Component (EMC) receives streams of raw event data and converts them into appropriate streams of events for the GUI, RMA and TDB. Each of these components receives a specific type of event stream: the RMA only receives relevant events encoded as Jess-formatted triples; the TDB receives all events in the form of OWL triples; the GUI receives relevant events in the form of object-attribute instances. The raw input streams are expected to be annotated using an event ontology with references to objects defined in the core ontology and the appropriate domain ontology. The event ontology currently being used in SAWA is shown in Fig. 5. This event ontology is known only to the EMC which converts all event information into appropriate structures for the other components; the isolation of the other components from the event ontology was done so as to permit the use of other event ontologies dependent upon the source of the event streams (which at this time is a simulator of fused object-level data).

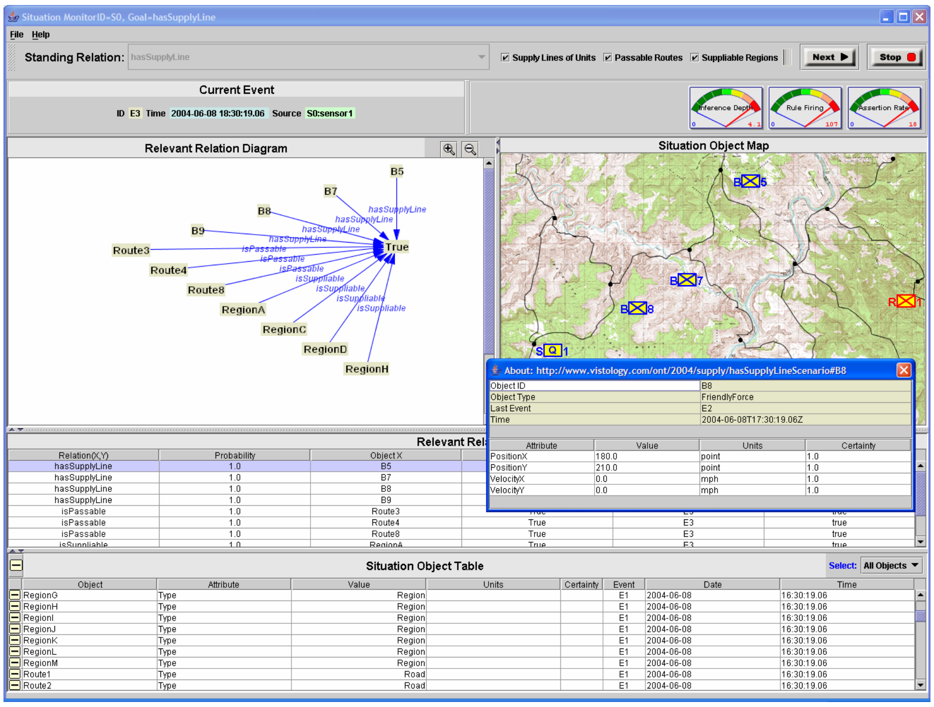
5.3 Relation Monitoring Agent

The Relation Monitoring Agent (RMA) performs the task of monitoring the stream of relevant events and detecting the truth value of relevant relations that might exist between objects occurring in the evolving situation. The RMA performs this task using the relevant rules defined by the domain knowledge in conjunction with the standing relation. These relevant rules are converted from their SWRL representation into Jess rules using an XSLT script. The Jess rules are then processed in the forward-chaining Rete network of our enhanced Jess inference engine; some of the enhancements we have made to Jess include the support for over thirty of the SWRL built-ins which are implemented as procedural attachments in the form of Java method calls. As events come in, they are processed through the Rete network and as a result may end up firing one or more rules. The firing of a rule results in the instantiation of a relation that is then reported to the GUI via the SMC. At the moment all rule firings result in relations that have an associated certainty rating of 1.0 (i.e., 100%). We are working on a new implementation of the reasoning engine that will incorporate uncertainty reasoning and will thus afford the detection of relations having incomplete certainties.

5.4 Triples Database

In RDF and OWL all information is represented in the form of triples. Each triple represents a predicate that relates a subject to an object. For example, to state that S2 is a SupplyStation requires a triple of the form: S2 rdf:type SupplyStation. More complex knowledge structure can be represented using collections of interrelated triples [12]. The triples representing the domain knowledge, user input and the incoming events all need to be maintained in a way that they can be readily processed. In SAWA this is accomplished through the Triples DataBase (TDB).

The TDB's primary purpose is to maintain an accurate history of all events so that they can be queried by the user at any time. It is currently developed on top of Jess and makes use of Jess' built-in query capabilities to implement an engine for OQL: OWL Query language [13]. The TDB also supports "what-if" queries in which a set of hypothetical facts are asserted, a query is run to produce what-if results, and the hypothetical facts are retracted along with all facts deduced from them. The TDB accomplishes this what-if capability using the "logical" retraction feature of Jess. While both the general query mechanism and the what-if query mechanism work as designed, they are quite inefficient and not particularly suited for new real-time operations. Consequently we are in the process of developing our own inferencing and query engine optimized for the processing of triples.



Current Event

ID E3 Time 2004-06-08 18:30:19.06 Source S0:sensor1

About: http://www.vistology.com/ont/2004/supply/hasSupplyLineScenario#B8

Object ID	B8
Object Type	FriendlyForce
Last Event	E2
Time	2004-06-08T17:30:19.06Z

Attribute	Value	Units	Certainty
PositionX	180.0	point	1.0
PositionY	210.0	point	1.0
VelocityX	0.0	mph	1.0
VelocityY	0.0	mph	1.0

Relation(X,Y)	Probability	Object X			
hasSupplyLine	1.0	B5			
hasSupplyLine	1.0	B7			
hasSupplyLine	1.0	B8			
hasSupplyLine	1.0	B9			
isPassable	1.0	Route3	True	E3	true
isPassable	1.0	Route4	True	E3	true
isPassable	1.0	Route8	True	E3	true
isSuppliable	1.0	RegionA	True	E3	true

Situation Object Table

Object	Attribute	Value	Units	Certainty	Event	Date	Time
RegionG	Type	Region			E1	2004-06-08	16:30:19.06
RegionH	Type	Region			E1	2004-06-08	16:30:19.06
RegionI	Type	Region			E1	2004-06-08	16:30:19.06
RegionJ	Type	Region			E1	2004-06-08	16:30:19.06
RegionK	Type	Region			E1	2004-06-08	16:30:19.06
RegionL	Type	Region			E1	2004-06-08	16:30:19.06
RegionM	Type	Region			E1	2004-06-08	16:30:19.06
Route1	Type	Road			E1	2004-06-08	16:30:19.06
Route2	Type	Road			E1	2004-06-08	16:30:19.06

Fig. 6. The SAWA GUI

5.5 Graphical User Interface

The Graphical User Interface permits the user to define standing relations, execute queries and monitor the current state of events, objects, attributes and relations. Its use on a Supply Logistics scenario (described in the next section) is illustrated in Fig. 6. The GUI provides the means for specifying the standing relation (i.e., goal), executing queries, and monitoring the evolution of events, objects, attributes and relations. Objects and attributes are displayed in the Situation Object Table and also on the Situation Object Map. Relevant relations appear in the Relevant Relations table as well as in the Relevant Relation Diagram. Clicking on objects on the map or events, objects or relations in the tables brings up a sub window of supplemental information as shown in the figure for Unit B8. The dials in the upper right hand corner are used for monitoring the performance of the inferencing engine.

6 A Supply Logistics Scenario

SAWA is currently being applied to the domain of supply logistics for which we have developed two scenarios, supply line and repairable assets. In this section we focus on the first scenario that was constructed for the purposes of demonstrating the basic system functions. The goal or "standing relation" for this scenario is to constantly monitor the relation "hasSupplyLine" for all friendly units. A supply line is defined as the existence of a continuous path of roads under friendly control connecting a unit (e.g., B5, B6, etc.) to a supply station (e.g., S1). The specific layout for this scenario can be seen in the map display in the GUI screenshot in Fig. 6. Roads connect pairs of regions (their centroids indicated by solid dots). There are six friendly blue units (i.e., B5, B6, B7, B9 and S1), including one supply station (S1), and one unfriendly red unit (R1).

The screenshot in Fig. 7 shows the simple supply logistics ontology that goes along with this scenario. Note that all of the classes in this ontology are implicitly sub classes of the Object class in the SAW Core Ontology described in Section 2.2 – this is necessary for the domain specific ontology to work with the otherwise generic mechanisms of the SAWA Engine. Note also that this ontology is a gross simplification of what would be expected for a more complete ontology necessary to support more practical supply logistics scenarios (which the authors are currently working on). This ontology was created using ezOWL, which produced the screenshot shown in Fig. 7 as well as the OWL code used in the running of the scenario.

The rule set developed for this scenario is partially shown in the screenshot of RuleVISor in Fig. 3. These rules define that a unit *hasSupplyLine* if the unit is in a region that *isSuppliable*. A region *isSuppliable* if it *hasSupplyStation* and is *underFriendlyControl* or if it is connected to another region by a *Passable* road and that other region *isSuppliable*. A region is *underFriendlyControl* if it contains a friendly unit. A region *hasSupplyStation* if the region contains an object and that object is a supply station (note that this rather obvious sounding rule is an implication that cannot be readily captured in OWL alone).

To simulate the running of the scenario several snapshots where developed as OWL annotations to define the state of the world at sequential time slices. In each time slice one of the units was moved around in such a manner as to create changes in the set of relations that would hold true. These snapshots where then presented to a running SAWA application in which the user specified the standing relation to be *hasSupplyLine* as applied to all friendly units. The system correctly detected the standing relations that held true at each time slice and reported these back to the GUI which displayed them for the user; the GUI screenshot in Fig. 6 shows the display after a couple of time steps.

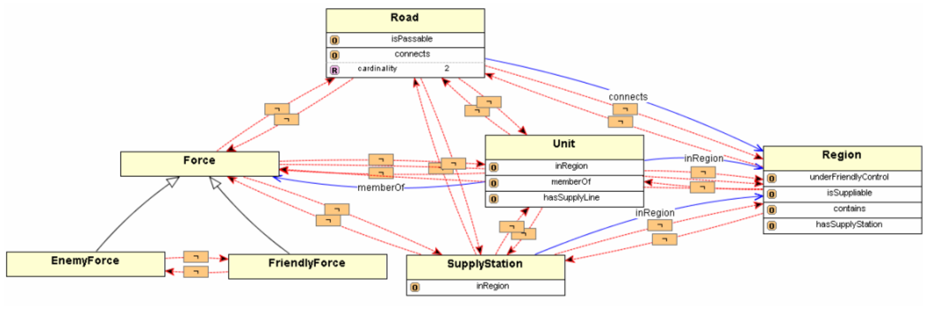


Fig. 7. Simple Supply Logistics Ontology. This ontology captures just enough information needed for reasoning about supply lines, which serves as the standing relation in our supply logistics scenario. Each of the classes represented in the ontology is a subclass of the Object class defined in the SAW Core Ontology shown in Fig. 1.

7 Semantic Web Technologies for Situation Awareness

The representational and reasoning requirements for Situation Awareness share much in common with those of the Semantic Web, with an added emphasis on the handling of time and uncertainty. Both need to be able to represent object-level information concerning classes and properties as well as higher-order relations that can occur among specific instances (e.g., a web site and its content, a web service and the set of users permitted to access it, etc.). Given our experience with using Semantic Web technologies it is natural to ask how well they fared when applied to Situation Awareness.

We have found the use of OWL to be generally quite suitable for representing taxonomies of classes and for capturing most of the properties of interest. There have been cases where we would have liked to have been able from within OWL to further constrain certain properties based on the values of other properties but instead were forced to use a rule. This is a well-known limitation of OWL [14] and is something that we have no problem with resorting to SWRL to resolve.

With regards to the use of SWRL there are a number of issues that we encountered (for more details see [15]). The lack of higher-order predicates is the most severe and was already illustrated in Section 2.2. Another issue we had to deal with was the declarative definitions of the SWRL built-ins. SWRL built-ins are defined without specification of the input/output nature of their terms. For example, swrlb:add(100,?X,?Y) is a perfectly valid use of the SWRL add built-in even though it defines an infinite set. It is also possible to use it in the following manner to implement subtraction, swrlb:add(100,50,?X), even though swrlb:subtract is also defined by the language. In practice "add" is generally needed as a function that binds to the variable in the first term position the summation of the remaining bound terms. In our implementation of the built-ins for Jess we require that there only be a single unbound variable and then use its occurrence in the list of terms to determine which function is to be used to calculate its value (this means you can, if you wish, use swrlb:add to perform subtraction); if more than one term is unbound an exception is thrown. This approach is not strictly conformant with the definition of SWRL but it represents a pragmatic approach that satisfies the requirements of a large number of problems.

Most rule languages have some mechanism for explicitly asserting new facts into working memory; in Jess this is achieved using (assert …). There is no such construct in SWRL. Rather, SWRL only states that when the statements in the body of a true are all true then the statements in the head are also true. The natural interpretation of this from the context of an inference engine like Jess is that the statements in the head should be asserted into working memory, for otherwise these true statements would be inaccessible by any of the other rules. For this reason we translate the statements in the heads of rules into assert commands in Jess. We go one step further in that we also look for the occurrence of variables in the head that are unbound in he body and produce a "gensym" command to generate a new symbol to produce an anonymous object to fill the role played by the variable. Technically the occurrence of a variable in the head that is not bound in the body is prohibited in SWRL (these are referred to as *unsafe* rules owing to the existentially quantified variables in their heads). In practice it is very frequently necessary to construct anonymous objects of this sort and yet SWRL has no construct for doing so (i.e., it has no gensym operator).

SWRL also lacks user-defined procedural attachments, which greatly reduces it general usefulness in practical applications. There are many calculations that are simply more appropriately handled by writing a method in Java (or any other procedural language) than to force its computation using rules alone. In situation awareness applications this comes up in such tasks as calculating the aggregation of a set of objects into a group, finding the centroid of a set of objects, dynamically modeling the position of a moving object over time, etc.

Perhaps the most restrictive aspect of SWRL is its lack of negation and in particular negation as failure. In all of our rule-based applications we have encountered the need to use a closed world assumption when reasoning about the information at hand. Seldom in the real world is it the case that we will have all of the timely information needed to make a conclusion; we must therefore be able to write rules that can detect the absence of specific forms of information and make decisions accordingly. In SWRL there is no way to look for the absence of information owing to its strict adherence to the monotonic assumption inherited from OWL. In our SWRL applications we have been forced to violate this assumption and move outside of the language in order to fully represent the knowledge needed to define some of our rules.

8 Conclusion

This paper described the Situation Awareness Assistant, SAWA. SAWA is designed to monitor the evolution of higher-order relations within a situation using formal and generic reasoning techniques for level-two fusion. The system was developed to make use of the formal languages of OWL and SWRL, which permit the representation of ontologies and rules. For a specific application of SAWA, a domain theory consisting of a domain specific OWL ontology and a corresponding set of SWRL rules are first constructed or reused from a previous application. A standing relation, or goal, is then defined by the user, which is used to determine the relevant portion of the domain knowledge for the current objectives as well as to identify the relevant object and object-attributes that the system needs to monitor in the event stream. As relevant events are detected they are passed on to the relation-monitoring agent, which analyzes them for the possible occurrence of higher-order relations. As higher-order relations are detected they are passed onto the GUI, which displays them in both tabular and graphical forms for the user along with other data pertaining to the events, objects and their attributes. The GUI also provides the capability for querying the system's triple database using basic OQL queries or with "what-if" queries that can produce hypothetical situations against which a query is run. A scenario from the domain of supply logistics was briefly described and we high-lighted some of the issues we encountered in our effort to apply Semantic Web technologies to the problem of Situation Awareness.

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. T. Berners-Lee, J. Hendler and O. Lassila, The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American, May 2001.
3. M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/
4. http://www.w3.org/2004/OWL/.
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. W3C Member Submission, 2004. http://www.w3.org/Submission/SWRL/.
6. C. Matheus, M. Kokar and K. Baclawski, A Core Ontology for Situation Awareness. In Proceedings of FUSION'03, Cairns, Queensland, Australia, pages 545-552, July 2003.
7. G. Klyne, J. J. Carroll, and B. McBride, Resource Description Framework (RDF) Concepts and Abstract Syntax.. W3C Recommendation 10 February 2004. Latest version is available at http://www.w3.org/TR/rdf-concepts/
8. European OntoWeb Consortium, A Survey of Ontology Tools, May 2002. http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip.
9. J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, S. W. Tu The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. 2002.
10. ConsVISor, 2003. http://www.vistology.com/consvisor. See also K. Baclawski, M. Kokar, R. Waldinger and P. Kogut, Consistency Checking of Semantic Web Ontologies. 1st International Semantic Web Conference (ISWC)}, Lecture Notes in Computer Science, LNCS 2342, Springer, pp. 454--459, 2002.

11. K. Baclawski, C. Matheus, M. Kokar, J. Letkowski and P. Kogut, Towards a Symptom Ontology for Semantic Web Applications. In Proceedings of Third International Semantic Web Conference, Hiroshima, Japan, pages 650-667, November, 2004.
12. RDF Primer. W3C Working Draft. Edited by F. Manola and E. Miller, 2002. http://www.w3.org/TR/rdf-primer/
13. OQL: OWL Query Language, 2003.
14. M. K. Smith, Web Ontology Issue Status, 2003. http://www.w3.org/2001/sw/WebOnt/webont-issues.html#I3.2-Qualified-Restrictions
15. C. Matheus, Position Paper: Using Ontology-based Rules for Situation Awareness and Information Fusion. W3C Workshop on Rule Languages for Interoperability, Washington, D.C., April 2005. http://www.w3.org/2004/12/rules-ws/paper/74