Situations and Policies

Brian Ulicny, Gerald M. Powell, Won Ng, Jakub Moskal, Mieczyslaw M. Kokar, *Senior Member, IEEE*

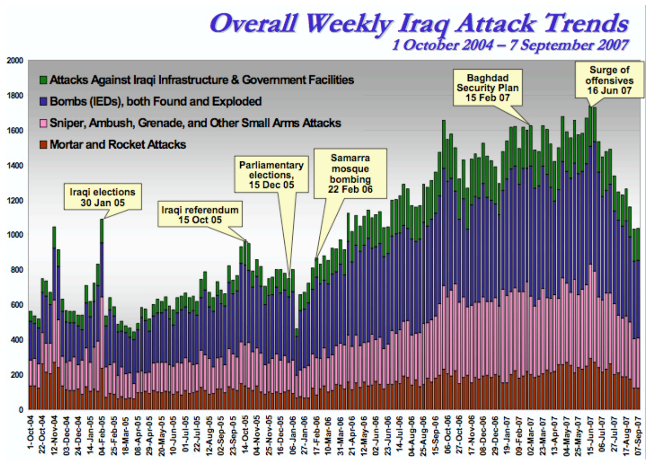
***Abstract*—In this paper, we first describe a system we have implemented that takes expressions of policies, expressed in a fragment of English called SBVR SE (Semantics of Business Vocabulary and Rules Structured English), an OMG standard, and automatically translates them into an executable semantic web formalism (OWL 2 and semantic web rules). Specifically, we describe how these policies can be used to automatically enforce compliance with policies and to reconcile multiple policies specified by independent parties. The scenarios implemented concern information sharing via XMPP ("instant messaging"). We then outline how situations can be characterized as policy-compliant or policy-violating. In some cases, situations are policy compliant or violating because of events and actions that they contain. We show that our formalism supports this analysis.**

***Index Terms*—Situation Theory; Actions; Events; Policies; Deontology; Rules**

I. Introduction

A "situation", at a minimum, includes some number of objects and relations among the objects during a time interval and at a place [18]. "Situation awareness" is the inference of a characterization of a situation based on a set of lower-level facts pertinent to that situation about which the agent has information. Situation awareness often takes the form of a statistical, quantitative inference. As an example, think of the characterization of the Iraq War in terms of the charts of various types of incident counts and other trends that Gen. David Petraeus and others presented in Congressional hearings (Figure 1). These charts depicted the situation in Iraq in terms of statistical trends among various indicators (relations): that IED incidents were on a sustained upward or downward trend or that Iraqi troops at a determined readiness level were increasing and so on. Progress was measured against baseline rates based on the start of the war or other milestones. In such cases, the characterization of the situation was not a

Figure 1 Chart from Petraeus, 2007 [15]



characterization into discrete logical categories, perhaps based on strict numeric thresholds. Rather, the characterization was mostly based on detection of strong statistical trends from period to period, toward or away from the baseline. As such, the idea of logically inferring situation types from lower-level situational data would, in general, seem to have limited practical application.

However, at least one type of characterization of situations does strongly lend itself to logical inference: the characterization of a situation as being compliant with or in violation of a policy. For example, it is either true or false that a set of financial transactions violates a set of anti-money laundering policies, although there may be some uncertainty as to which, based on incomplete or ambiguous information. The violation or non-violation of the policies is not a matter of degree. If the prohibited conditions exist, then the situation is a violation of the policy. Whether the legal or illegal behavior is increasing or decreasing is a different matter.

In this paper, we first describe a system we have implemented that takes expressions of policies, expressed in a fragment of English called SBVR SE (Semantics of Business Vocabulary and Rules Structured English) [1], an OMG standard, and automatically translates them into an executable semantic web formalism (OWL 2 and semantic web rules). We describe how these expressions can be executed automatically to enforce compliance with security policies they describe and to reconcile multiple policies by independent parties. As such, the policy engine can characterize the corresponding situations as policy-compliant or policy-violating.

We then discuss how situations can be characterized as

Manuscript received October 31, 2011. This research was performed under U.S. Army contracts W91260-09-C-0037 (USASMDC) and W15P7T-09-C-S031 (CERDEC I2WD). The title alludes to Barwise and Perry's book, "Situations and Attitudes" (MIT Press, 1983), the primary text of Situation Theory.

Brian Ulicny, Ph.D., VIStology, Inc, Framingham, MA., bulicny@vistology.com
Gerald. M. Powell, Ph.D., U.S. Army Research Laboratory, Aberdeen Proving Ground, MD. gerald.m.powell@us.army.mil
Won Ng, VIStology, Inc., Framingham, MA, wng@vistology.com
Jakub Moskal, Ph.D., VIStology, Inc., Framingham, MA, jmoskal@vistology.com.
Mieczyslaw M. Kokar, Ph.D., Northeastern University, Boston, MA. mkokar@ece.neu.edu

policy-compliant or –violating both because of the actions that they contain or independently of such actions. We extend our Situation Theory Ontology formalism to include actions and events as relevant individuals and show that this formal representation is adequate to encode relationships between situations, actions and policies and to derive inferences from them

II. Representing Policies Semantically

In this project we were concerned with the ability to use policies to ensure compliance during runtime as well as with the ability to do policy reconciliation. Policy compliance involves the run-time process of ensuring that all of the conditions defined by a policy hold true; a common example is the checking of credentials required before granting access to a document. In policy reconciliation, the goal is to take multiple polices and generate a policy instance that simultaneously satisfies all of them; a typical example here is determining specific conditions under which a communication session is to be established between nodes in a VPN where the ends of the connection are governed by different policies.

1.1 Semantic and Non-Semantic Representations of Policies

Policies can be implemented in a system via the hardware (e.g. this light will not turn on unless both of these switches are turned on); or in software. In software, a policy can be represented either syntactically or semantically. By a semantic representation, we mean a representation in which inferences can be made on the basis of a policy instance using a domain-generic inference engine. So, for example, a Windows Group Policy instance has a meaning that is clear to everyone who knows the semantics of the policy language. However, no generic reasoning engine can draw inferences from Windows Group Policy instances in their native format. The representation has no meaning to those engines. Instead, procedures need to be written to interpret the policies.

A primary objective in our work is to develop the means by which operations-governing policies can be handled automatically by a computer using a generic inference engine. For this reason it is important to be able to describe policies in a formal, declarative way that will permit them to be automatically processed by formal reasoning engines.

A formal reasoner or inference engine is a system capable of applying the formal axioms of a language to a body of data/facts/knowledge resulting in the derivation of additional inferable facts. A rule-based system, for example, may be used as a formal reasoner if it is provided with a set of axioms for the language in which the data/knowledge is represented. Such axiom sets are available for a number of ontology languages as discussed below.

An important principle employed by many systems including policy-based reasoners is the use of the closed world assumption (CWA), which permits systems to assume that everything that is known to be true of the "world" is available in the facts that have been provided about it; if a fact is not explicitly stated it is assumed to be false. The closed world defined by a set of facts can be thought of as a "context" in which reasoning is to occur. OWL-based systems, like the one we describe here, adopt the open world assumption. In a policy context, this means that the engine might not know everything relevant to determining whether a policy is violated or not.

For reconciliation to be possible there should be an explicit separation of policies and mechanisms that use the policies, and the policies should be first-class objects within the system about which the system can reason. In this way, policies will be objects that can be represented, stored and manipulated by the system. Moreover, in this way policies will have their own interpretation, or semantics. This has a very important impact on the accreditation process in that mechanisms can be accredited and then policies can be added dynamically.

III. Policy Languages

In our project, we used SBVR Structured English (SE) for authoring policies in an English-like formalism. SBVR SE policies are then automatically translated into proprietary BaseVISor Rule Language (BVR) for execution and policy reconciliation. Currently, we are also working on automatically translating SBVR SE to Rule Interchange Format (RIF) Core rules, which is a W3C published recommended standard [20].

1.1.1 SBVR Structured English

Semantics of Business Vocabulary and Business Rules (SBVR) [1] is an OMG standard introduced in 2008 that aims at a more natural format for expressing rules. Business rules are expressed in a subset of natural language that is readily understandable by business people, instead of at an implementation level, such as rules that are processable by a formal reasoning engine. The vocabulary represents the concepts used in the rules and can also express facts and relations between concepts (i.e. ontological relations). The specification is based on formal logic and captures the semantics of implementation-independent business models. SBVR is located in the Business Model (also called the Computation-Independent Model) level in OMG's Model Driven Architecture (MDA) [2] and is meant to be translatable to a Platform-Independent Model (PIM) that describes the structure and behavior of the model, and subsequently to a Platform-Specific Model (PSM) that includes all the platform dependent information necessary for a developer to implement executable code, such as specific programming language packages. SBVR is mapped to the Meta-Object Facility (MOF) [3] metamodel – a useful feature for transformations of an SBVR model to other models.

SBVR distinguishes between *alethic* and *deontic* rules. Alethic rules are categorized as structural business rules, which are rules that must necessarily be true as part of the business organization. For example, the rule that "it is necessary that all supervisors be full-time employees" expresses an alethic contraint. Deontic rules are operative business rules that are expected to be obeyed but can be violated in practice. For example, "it is obligatory that employees fill out a W-2 form." This rule expresses an obligation that is supposed to be observed, but may actually be violated in practice.

SBVR has two common notations: Structured English and RuleSpeak®. We do not discuss the RuleSpeak notation here. SBVR Structured English (SBVR SE) is a controlled English vocabulary and grammar that uses font styling and colors to indicate SBVR concepts. term represents a noun concept such as rule and action. Name is an individual concept and usually is a proper noun, e.g. California. *verb* is part of a SBVR construct called a fact type and is usually a verb, preposition or combination of preposition and verb. Lastly, SBVR SE defines a set of keywords that are reserved words or phrases with special meaning. Examples of keywords are the articles a and the, modality phrases it is necessary that, and quantifications every and at most one. An example of a SBVR SE rule is:

It is obligatory that a driver *is qualified* if the driver *rents* a car that *is owned by* EU-Rent

Like the other languages discussed, SBVR is domain and application independent. The SBVR specification includes a proposal relating SBVR concepts to equivalent OWL expressions, so clearly some consideration was given to how SBVR should work with semantic languages. Its main strength over the other languages is its user friendliness. Because SBVR SE is an almost-natural language, it is suitable for expressing high-level rules.

SBVR is sufficiently expressive for representing high level rules but because SBVR is at the business model level, it suffers from the common problem that most business model level components do: translation to a PIM and especially to a PSM requires additional details about computations and platform-specific information, usually supplied by an IS or IT person. The SBVR vocabulary can be expanded to include platform vocabulary and RuleSpeak includes templates to write computation rules, but SBVR is meant to be a high level language and is not executable, so SBVR is only truly useful when translated into a lower level executable language like BVR or Rule Interchange Format (RIF)-Core.

A. *BaseVISor Rule Language (BVR)*

BaseVISor (http://www.vistology.com/basevisor) is a versatile forward-chaining rule engine specialized for handling facts in the form of RDF triples (i.e., subject, predicate, and object). It expresses rules in BaseVISor Rule language (BVR). The engine implements OWL 2 RL inference rules and supports XML Schema Data Types.

Generally speaking, rules are expressed in the form of if/then statements. The ‘if’ part of the statement is represented by the ‘body’ or ‘antecedent’ of the rule; the ‘then’ part is represented by the ‘head’ or ‘consequence’. In BVR the contents of rule heads and bodies are made up of triple patterns (i.e., triples that may contain variables) and procedural attachments, i.e. functions such as add, assert, and println (print line). Users can add user-defined procedural attachments for use in rules. BaseVISor also supports BVR queries, which are special cases of rules with empty heads, and are useful for retrieving information from the resulting fact base.

BVR is domain and application independent, compatible with the semantic languages OWL and RDF, designed for formal reasoning and executable in the BaseVISor environment. It is very expressive, especially since the language is extensible via user-defined procedural attachments.

B. *RIF-Core*

The *Rule Interchange Format* (RIF) is a W3C Recommendation. Since there are many rule languages in existence, RIF is envisioned as a lingua franca in which rules can be exchanged by expressing them in a common language. RIF-Core is a subset of the Rule Interchange Format. RIF-Core corresponds to the language of definite Horn rules without function symbols (often called 'Datalog') with a standard first order semantics. RIF-Core thus is a subset of RIF-BLD (Basic Logic Dialect). At the same time, RIF-Core is a language of production rules where conclusions are interpreted as assertion actions. Thus RIF-Core also is a subset of RIF-PRD (Production Rules Dialect). RIF-Core is based on built-in functions and predicates over selected XML Schema datatypes, as specified in RIF-(Databyes and Built-ins) DTB 1.0.

To give an example, the SBVR SE rule:

It is necessary that a customer has_status Gold if the customer has_status Silver and the customer has_shopping_cart a shopping_cart that has_worth a value that is_greater_than $2000.

corresponds to the RIF-Core rule:

```
Forall ?customer ?shoppingCart (
         ?customer[ex1:status->"Gold"]
         := And(
       ?customer # ex1:Customer
       ?customer[ex1:status->"Silver"]
       ?shoppingCart # ex1:ShoppingCart
       ?customer[ex1:shoppingCart->
                      ?shoppingCart]
       ?shoppingCart[ex1:value->?value]
       pred:numeric-greater-than-or-
       equal(?value 2000)
       )
  )
```

C. *Translation of SBVR SE to BVR and RIF-Core*

For the automatic translation from SBVR to BaseVISor, we use ATL (Atlas Transformation Language) Given the metamodels of source and target languages, ATL transforms a model conforming to the source language metamodel to the target language. We have created a metamodel for BaseVISor and modified the SBVR metamodel used in a UML-to-SBVR effort [6]. We then wrote ATL rules to map from SBVR rules expressed in Structured English to BaseVISor rules. Because BaseVISor Rule Language is not a standard, we are in the process of doing the same thing for RIF Core.

IV. Policy Ontologies

We developed OWL ontologies to encapsulate our treatment of policies and to represent concepts and their relations that we have determined to be essential for policy compliance and reconciliation scenarios, including information exchange policies. These "core" ontologies are the basis for any domain-specific application of policy reasoning, i.e., domain-specific scenarios should extend these ontologies with their domain-specific knowledge and rules. The design of the ontologies, such as treating actions and operations as first-class entities, are grounded in our study and investigation of formal security models, which involve policies.

A. Representing Modal Notions in OWL

Our system, called PolVISor involves both forms of modality, both deontic and alethic. Modal expressions qualify the truth of a statement. For example, to say that "John is possibly dyslexic" is not to assert that "John is dyslexic", but a more qualified statement that the statement might be true. Modality is expressed logically as operators over propositions. *Op(p)* means that some modal operator *Op* is being asserted of the proposition p: *It is Op that p*. The operator identifies the way in which the truth of the bare proposition *p* is being qualified.

Alethic modality is the logic of possibility (it is possible that *p*) and necessity (it is necessary that *p*). As specified by SBVR, alethic notions are encoded directly in the ontology. Necessity relations between classes are expressed in terms of subclass relations that apply to all instances. Thus, to say that necessarily, all bachelors are unmarried is to say that the class Bachelor is a subclass of Unmarried Things. Without such a subclass relation, it might be the case that all of the instances of Bachelor are instances of Unmarried, but that would be a contingent coincidence, not a necessary truth, with respect to that ontology. We encode that it is possible for a member of class *F* to be a member of class *G* in the ontology by failing to have classes *F* and *G* as disjoint classes. If *F* and *G* are marked as disjoint classes, then necessarily, no *F*s are *G*s, (and, necessarily, no *G*s are *F*s), according to that ontology.

"It is necessary that a user has a password" expresses a necessary relation between the class of Users and the class of things that have a password. This necessary relation would be expressed by saying that the class of Users is a subclass of the class of things that have Passwords. This encodes the necessity relation in the ontology directly. Ontologies express constraints on how the world can be. To say that users *may* have a password is expressible by saying that the class of Users and the class of things that have a password are not disjoint.

Deontic Logic [5] is the study of the logic of the concepts "may" (or deontic 'can') and "must" and their duals "may not" and "must not". These concepts are crucial in expressing policies: policies express what may or may not be done, under certain conditions, and what must and must not be done, again, under certain conditions. *May* and *must* are modal notions. Sentences employing modal notions, like may and must, do not express the way the actual world is, but qualify the truth of the proposition they modify, in this case expressing conditions on how possible worlds must be if they are to comply with the deontic notions our ontology encodes. That is, if I say that "John may go to the store" or "John must (or must not) go to the store", I do not say anything about how the actual world is with respect to John's going to the store. What I express has to do with the consistency of John's going to the store with the ways in which John is permitted to act or with the ways in which John must act.

In our inference engine, BaseVISor, propositions are expressed as RDF or OWL triples (subject, predicate, object). BaseVISor does not allow for modal operators over triples. Therefore, rather than give modal operators their usual semantics as quantifiers over possible worlds or ways the world could be or ways a person could act, we treat Actions as a class that can be subdivided into Permissible (may), Omissible (may not), Optional (may and may not), Obligatory (must) and Prohibited (must not) subclasses.

The structure of the ontology is represented in Figure 2:

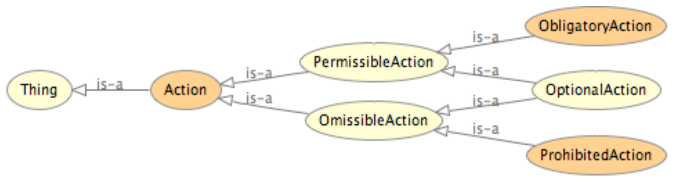


Figure 2. Classes and subclasses of Deontic Ontology

First, Actions are subclassified as Permissible or Omissible.

An action is Permissible if it may be done. For example, if getting married is permissible (without restriction), then the class of actions that are Marriages could be represented as a subclass of the class of permissible actions.

An action is Omissible if one may not do it. For example, eating okra is omissible. One may abstain from eating okra. The class of actions that is okra-eating could thus be represented as a subclass of the Omissible actions.

In fact, one both may and may not eat okra (and one may or may not get married), so both of these classes of actions would be subclasses of the intersection of the Omissible and Permissible classes: the Optional actions.

Obligatory actions (actions one must do) are a subset of the Permissible actions. If an action must be done, then it may be done. The Obligatory actions and the Omissible actions are disjoint: if an action must be done, it is not the case that it may not be done.

Similarly, Prohibited actions (actions one must not do) are a subset of the Omissible actions (actions one may not do). The Prohibited actions and the Permissible actions are disjoint: if an action must not be done, then it is not the case that it may be done.

We have expressed these relations in an OWL ontology. The ontology may be downloaded at http://vistology.com/ont/2010/secpol/Deontic.owl.

By means of this ontology, one can state that all instances of actions of a certain type are, for example, prohibited (e.g. theft, murder) or permissible (e.g. expressing one's opinion, forming associations) across the board. Policy rules allow one to express conditions under which actions of a certain type are classified as permissible or prohibited or optional based on additional facts about them. For example, one could express the policy that it is permissible to marry only if one is at least a certain age, not currently married, and so on.

Because of the open world assumption in OWL, the fact that an action is obligatory does not lead to the inference of a policy violation if there is no such action yet. Stating that an action is obligatory entails that there must be such an action at some point. We can infer a contradiction if an action is asserted to be both obligatory and omissible, however. Further, we can say that if an action must be completed within a certain time frame, if the time frame contains no such action, then this is a violation.

In the next section, we illustrate our approach by showing how certain information sharing actions can be inferred to be policy violating or compliant via formal inference.

V. XMPP Information Exchange Policies

Extensible Messaging and Presence Protocol (XMPP) [6] is a popular open-standard protocol for instant messaging (IM) widely used in military applications. There are a number of extensions to the protocol that define protocols for other functionality, for example Voice Over IP (VoIP) can use XMPP for internet telephony.

Each user signs into his XMPP account identified by a jid, commonly of the form *name@domain.server*, e.g. *juliet@montague.net*. Each jid has a contact list called a roster. The server hosting the user automatically sends a presence to each of his contacts, except for those he has blocked, to indicate that he is now online. The contact's server forwards the presence to the receiver, unless she specified that she does not wish to receive presences from the sender. The contact's server also sends back a presence to the sender if she has not blocked presence-outs to the sender. Now the two clients can start chatting with each other. Users can also join chatrooms, participate in conversations as a group, and send messages to individuals in the room.

Using Openfire [7], an open source XMPP server available from Ignite Realtime [8], for our server, we developed a plugin that intercepts incoming and outgoing XMPP stanzas. The stanzas of interest in our scenarios are presences and messages, but all stanzas are intercepted so our implementation is extensible. Users connect to the servers via Spark IM Client [9], an open source IM client application also provided by Ignite Realtime.

We wrote policies in SBVR SE corresponding to the scenarios outlined below and converted them to an executable rule language (BVR). We installed the policies on XMPP servers and used them to control who could talk to whom according to the policies and making use of descriptions of the users encoded in either the friend-of-a-friend (FOAF) or vCard vocabularies for describing persons.

To demonstrate server policies that limit who can communicate with whom, based on facts about the persons involved, we developed the rules and ontologies for one server that restricted chat based on gender and another server that restricts chat based on the first letter of the jid. The first server allowed men to be visible Monday, Wednesday, Friday, and women the other days. The second server, allowed jids that began with the first half of the alphabet to be visible the first half of the week; the remainder, the second half. The policy engine correctly interpreted and enforced the policies on each server, prohibiting messages and presences from being sent when the users were not compliant with the policy on that day.

In a second scenario, we implemented security level markings for jids. We added policies limiting communications to those with the appropriate security level. For example, someone with a clearance of Top Secret could send and receive messages classified as Top Secret or any "lower" level like Secret or Unclassified. In this scenario, the policy engines correctly prevented communications between users involving illicit classification levels.

In both of these scenarios, there was no policy reconciliation involved. In order for both servers to comply with all of the applicable policies, it was only necessary for each server to enforce its own policies on incoming and outgoing messages and presences.

In a third scenario, we implemented explicit reconciliation of policies between servers. In this scenario, both Server 1 and Server 2 had security policies restricting who can join what chatroom. Their policies must be successfully reconciled and the attempt to join must satisfy the reconciled policy in order for the attempt to be allowed. By satisfying the reconciled policy, the request also satisfies each server's policy. The servers exchange policies, compute a reconciled policy, and implement it. We were able to successfully compute a reconciled policy, propagate it to the other server, and enforce it.

In a fourth scenario, we implemented the same policy reconciliation scenario but with each server expressing facts about its users in a different ontology (using vocabularies corresponding to the FOAF and vCard standards) that must be aligned automatically before reconciliation takes place.

With these scenarios, we demonstrated that:

1. Policies authored in a restricted natural language format (SBVR Structured English) can be automatically converted to an executable formalism (BaseVISor rule language and OWL 2 RL) effectively.
2. Policies written in the ontology-based rule language can provide an effective and flexible way to specify expressive policies regulating actions that can be automatically enforced using ontology-based reasoning. The core ontologies used as the basis for domain-specific knowledge are grounded by our investigation of established security models.
3. Policies written in the ontology-based rule language can be effectively reconciled to allow for dynamic, policy-based information exchange between an open set of XMPP servers.
4. While policy reconciliation typically requires the sharing of a common vocabulary, we have shown that effective ontology matching can be implemented to allow policy reconciliation across different (but similar) vocabularies.

Video demos of these scenarios can be viewed at: http://173.14.188.57:9999/secpol/

VI. Situations

In the previous section, we illustrated formal reasoning about constraints on actions imposed by policies. In this section, we extend the analysis of policy violation and compliance to the notion of a situation that contains or involves an action constrained by a policy. We do this by extending an ontology we have developed previously to include actions as individual entities. Doing so allows us to extend the notion of policy compliance from the level of an individual action to the context (situation) in which it occurs.

Situation Theory, as initiated by Barwise and Perry [16] and developed by Devlin [17], is a theory of information flow among cognitive agents, particularly by means of language. Barwise and Perry begin with the assertion that people use language to talk about (i.e., exchange information about) limited parts of the world, which they call situations. (For example, scenes are situations that are visually perceived by some observer.) Abstract and concrete situations are partial possible worlds, and the information an agent has about a given situation at any moment is limited to information about elements of the situation.

In situation theory, information about a situation is expressed in terms of infons. Situations support (|=) infons. Infons are written as

$<<R, a_i, \ldots, a_n, l, t, 0/1>>$

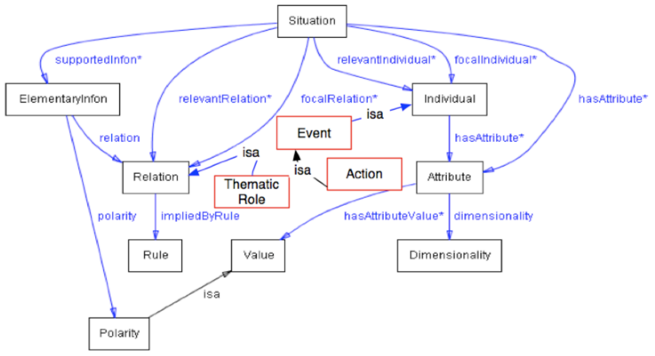
where R is an n-place relation and $a_1, \ldots, a_n$ are objects appropriate for R. Infons have slots for time and location parameters, which, by convention, are encoded as the two slots before the final polarity parameter (0 or 1). Each slot is associated with a type of individual that must fill it (e.g. times, locations, persons). A polarity of 1 indicates that the situation contains the described state of affairs. A polarity of 0 indicates the opposite. Infons may be recursively combined to form compound infons.

In [18], a (partial) computer-processable implementation for Situation Theory was developed that is compatible both with Barwise and Perry and with Endsley's model of human situation awareness [19]. To achieve this, Situation Theory was encoded using a formal ontology (STO) in OWL (Figure 3, excluding red boxes). An ontology-based approach to situation awareness supports the inference of new facts about the situation from the encoded facts.

It has been shown that the OWL ontology encoding Situation Theory can be used to model and track situations as they unfold. Implicit features in Situation Theory notation are made explicit in the STO OWL notation.

Similarly to Barwise and Perry, we consider situations to be associated with spatiotemporal regions, although STO does not explicitly identify the class for location or time; these are just special types of Attribute. Endsley defines Situation Awareness as "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future." That is, for Endsley, situation awareness is not the perception of some thing, a situation. Rather, it is a perception of "what is going on" with entities in a spatiotemporal region and a projection of their status in the future, in order to support decision making.

Figure 3 Situation Theory Ontology (STO) extended to include Events, Actions and Thematic Roles



Situations evolve, and by this we mean that situations exhibit differences across time, just as they exhibit differences across space. Situations have an associated spatiotemporal extent.

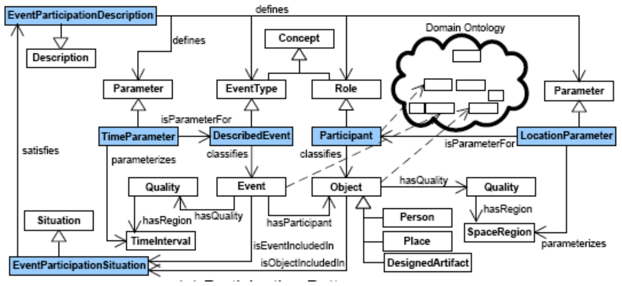
Clearly, situations can have discontiguous spatiotemporal regions: for example, a phone conversation is a situation that takes place at two discontiguous locations. Similarly, a play with an intermission can be thought of as a situation with a discontiguous time span.

How can we represent that a situation complies with a policy or violates a policy?

Our Situation Theory Ontology does not contain a class of events. Therefore, what is the relation between our situation ontology and the event-based ontology that our policy reasoning employs? (Actions are events with agents; they are events that someone makes happen, as opposed to events that simply occur, like rain or snow.)

In order to bridge the gap between the two vocabularies, one involving events and actions over which policies are defined, and the other involving situations over which situation awareness is defined, for the purposes of this paper, we consider events to be a subclass of Individual within a situation. Actions are a subclass of Event: an Event that has something as an agent. For the purposes of this paper, we use the "F" event model ontology ("Event Model F") described in [23]. This ontology is based on the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) (Figure 4). See [24] for a comparison of similar ontologies.

Figure 4 Event Model F (diagram from [23])



We depict the extended Situation Theory Ontology in Figure 3. The class of Event (from Event Model F) is made a subclass of Individual. The class of Thematic Role is a subclass of Relation. Thematic Roles have domain Event and range Individual. They include *agent, theme, instrument, source, goal,* and so on. Thematic roles are ways in which individuals participate in events.[1] For example, something that bears the *agent* relation to an event is the participant who brings about the action, and something that bears the *theme* relation to the event is the entity that undergoes the action or event. Something that bears the *goal* relation to an event is the participant at which (to whom) the action is directed, as in John sent a message *to Mary*.

To illustrate, we encode a situation in which John joins Chatroom 1 at t, John sends a presence to Mary at t+1 and John sends Message M to Mary at t+2 as follows (suppressing location, for simplicity):

s |= <agent,e1, John, t, 1>
s |= <theme,e1, Chatroom1, 1>

where e1 is of rdf:type Join, and Join is rdfs:subclassOf Event

s |= <agent, e2, John, t+1, 1>
s |= <theme, e2, Mary, t+1, 1>

where e2 is of rdf:type SendPresence, and SendPresence is rdfs:subclassOf Event.

s |= <agent, e3, John, t+2, 1>
s |= <theme, e3, Message_M, t+2, 1>
s |= <goal, e3, Mary, t+2, 1>

where e3 is of rdf:type SendMessage, SendMessage is rdfs:subclassOf Event.

The Relations agent, theme and goal are Thematic Roles, which are a subclass of Relation, with domain Event and range Individual. In this way, we represent that several events take place within the context of a single information exchange situation.

In addition to allowing us to express the relation between situations and actions, this also has the virtue of allowing us to more explicitly encode n-ary relationships within the STO ontology. In the STO ontology, argument places are denoted simply by anchor1, anchor2 and anchor3 object properties. These do not have any explicit meaning. To say that Bob is the first argument of the Chase relation and John is the second argument, it is not clear who is the agent of the chasing and who is being chased. The domain and range of the Chase relation do not help us to distinguish the roles here. In the extended ontology, we represent this as follows (ignoring time and location for now):

s |= <Agent,e,Bob,1>, <Theme,e,John,1>

[1] Thematic Role. Utrecht Dictionary of Linguistics. http://www2.let.uu.nl/UiL-OTS/ Lexicon/zoek.pl?lemma=Thematic+role &lemmacode=150

where e is of rdf:type Chase, a subclass of Event.

Secondly, we can encode a taxonomy of action types. For example, we can say that Chase is a subclass of Move. Therefore, we can infer that if Bob is the agent of an action that is a Chase, then he is an agent of a Move action (a motion).

Finally, we can encode relations of arbitrary arity in an explicit way. For example, to say that a situation involves John sending Mary in London a book via Fedex, we can assert:

s |= <agent,e,John,1>,
s |= <theme,e,book,1>,
s |= <goal,e,Mary,1>,
s |= <in, Mary, London, 1>,
s |= <instrument,e,Fedex, 1>,

where e is of rdf:type Send.

A situation contains an action if it supports a positive infon in which the action appears.

We can now define when an action performed in a specific situation is compliant with a set of policies. If an action is inferred to be forbidden by a policy, then the action violates the policy. Every situation that contains an action that violates a policy is noncompliant with that policy. So, for example, suppose that John's sending Mary his presence is forbidden by a policy (as in one of the scenarios described above) because of John's gender and the day of the week. (The policy of one server states that men can only send presences only on Mondays, Wednesdays and Fridays). As such, if it is a Tuesday, the action e2 is forbidden by the policy. Thus, action e2 is policy non-compliant, and therefore, situation s which supports a positive infon that contains e2 is also policy non-compliant.

A corresponding action and corresponding situation would be policy compliant in respect to the policy described if it had taken place on a Monday, Wednesday or Friday, or if a female had sent the presence on Tuesday. Thus, actions of the same action type and situations that are identical in most respects can be policy-compliant or policy-violating depending upon the circumstances and the individuals involved.

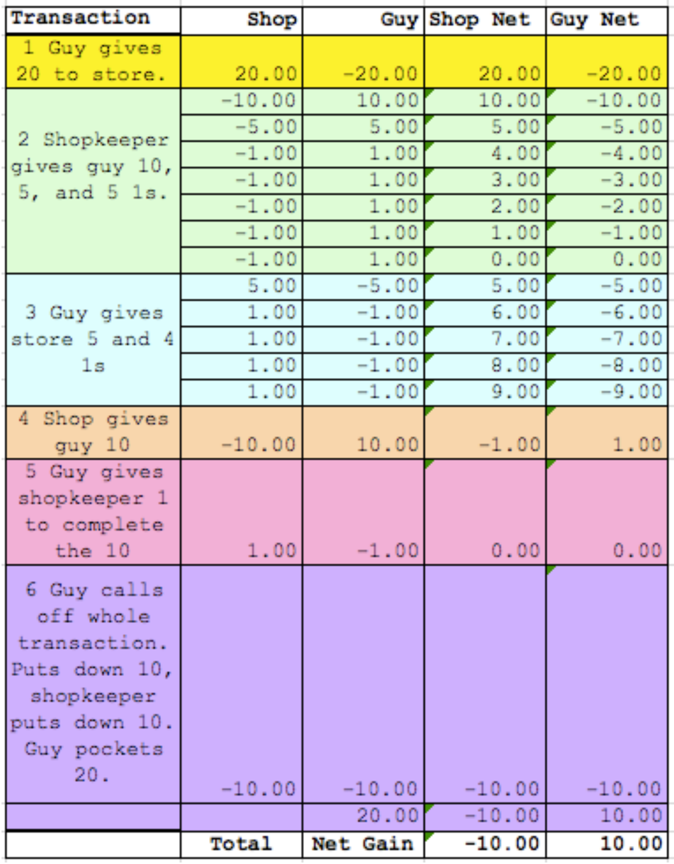
A situation, expressed in the STO vocabulary, is policy-compliant if it contains no actions that violate a specified set of policies, in the sense of supporting no positive infons that involve forbidden events

By means of this encoding of the situation or context that an action occurs in, we can also encode rules or ontological constraints on actions that occur in a context in which a forbidden action occurs or in which an obligatory action occurs. For example, if an exchange of information violates security levels, then all subsequent exchanges of information in that context are tainted.

In other situations, we can say that the entire situation involving a number of actions is policy violating even without knowing what action was forbidden.

This can be illustrated by the scam known as "change raising". Videos illustrating the scam may be readily found online. In one such clip ("The Change Rasising Con"), from the BBC program “The Real Hustle”, the scammer first starts a transaction of getting change for a 20 pound note. Then, he exchanges some of the smaller bills for a 10. Finally, he ends up calling the whole transaction off, ending up with 10 pounds more than he started The steps are illustrated in Table 1. Positive numbers represent receipts. Negative numbers represent disbursals. The conman nets 10 pounds by misguiding the shopkeeper, who doesn’t realize her mistake.

Table 1 Change-Raising Scam, broken down into component actions



Transaction	Shop	Guy	Shop Net	Guy Net
1 Guy gives 20 to store.	20.00	-20.00	20.00	-20.00
2 Shopkeeper gives guy 10, 5, and 5 1s.	-10.00	10.00	10.00	-10.00
	-5.00	5.00	5.00	-5.00
	-1.00	1.00	4.00	-4.00
	-1.00	1.00	3.00	-3.00
	-1.00	1.00	2.00	-2.00
	-1.00	1.00	1.00	-1.00
	-1.00	1.00	0.00	0.00
3 Guy gives store 5 and 4 1s	5.00	-5.00	5.00	-5.00
	1.00	-1.00	6.00	-6.00
	1.00	-1.00	7.00	-7.00
	1.00	-1.00	8.00	-8.00
	1.00	-1.00	9.00	-9.00
4 Shop gives guy 10	-10.00	10.00	-1.00	1.00
5 Guy gives shopkeeper 1 to complete the 10	1.00	-1.00	0.00	0.00
6 Guy calls off whole transaction. Puts down 10, shopkeeper puts down 10. Guy pockets 20.	-10.00	-10.00	-10.00	-10.00
		20.00	-10.00	10.00
	Total	Net Gain	-10.00	10.00

Perhaps no known policy forbids any action on the shopkeeper’s part or the scammer’s part. Nevertheless, the entire situation might be seen as violating a policy: the policy that change-making situations, in which cash is exchanged only for cash, should not end up with a net loss. Thus, the situation is policy-violating, even though it is not clear which action within the situation is prohibited.

VII. Conclusion

In this paper, we have shown how policies can be expressed in SBVR SE and translated into an executable rule language that can constrain information exchange actions in XMPP scenarios. We then went on to show how our Situation Theory Ontology could be extended to include actions and events as individuals. This enabled us to extend formal reasoning about policy compliance to situations as a whole, which we illustrated with various examples. We showed that situations may be inferred to be policy compliant because of actions in infons they support. Policy violations within a context can then affect the status of other actions. We also outlined a case in which situations can be policy (non-) compliant based on the cumulative effect of several actions.

References

[1] Semantics of Business Vocabulary and Business Rules v1.0. http://www.omg.org/spec/SBVR/1.0/ . May 2011

[2] MDA Home Page. www.omg.org/mda/ . May 2011.

[3] MOF Home Page. www.omg.org/mof/ . May 2011.

[4] Lee, R. Using New Standards to Develop IC Ontologies. In Proc. of the Fifth International Conference on Semantic Technologies for Intelligence, Defense, and Security. (STIDS’10). Fairfax, VA, USA, October 27-28, 2010.

[5] P. McNamara, Deontic Logic. The Stanford Encyclopedia of Philosophy (Fall 2010 Edition), Edward N. Zalta (ed.). http://plato.stanford.edu/archives/fall2010/entries/logic-deontic/

[6] XMPP Standards Foundation. www.xmpp.org/ . May 2011.

[7] Openfire http://www.igniterealtime.org/projects/openfire/ . May 2011.

[8] Ignite Realtime Home Page. http://www.igniterealtime.org/ . May 2011.

[9] Spark Home Page. http://www.igniterealtime.org/projects/spark/index.jsp . May 2011.

[10] CWID 2010 UK Cross Domain Chat. Enclosure 1 to Cross Domain Chat Point Brief. July 2010.

[11] Boldon James – Military Messaging and Secure Information Exchange Software Page. http://www.army-technology.com/contractors/navigation/boldonjames/ . May 2011.

[12] Isode Whitepaper: Using Security Labels to Control Message Flow in XMPP Services. http://www.isode.com/whitepapers/controlling-message-flow.html . May 2011.

[13] Common Information Sharing Standard for Information Security Marking: XML Implementation Implementation Guide. Office of the Director of National Intelligence Chief Information Officer. Release 2.0.3, February 2006.

[14] J. Euzenat, F. Scharffe, and A. Zimmermann, “Expressive alignment language and implementation,” deliverable, Knowledge Web NoE, 2007. Available at http:// ftp//ftp.inrialpes.fr/pub/exmo/reports/kweb-2210.pdf

[15] Gen. David H. Petraeus, Report to Congress on the Situation in Iraq, Joint Hearing of the House Committee on Foreign Affairs and the House Committee on Armed Services. Sept 10, 2007.

[16] J. Barwise, J. Perry, Situations and Attitudes, MIT Press, 1983.,

[17] K. Devlin, Logic and Information, Cambridge U. Press, 1991.

[18] M. Kokar, C. Matheus, and K. Baclawski, "Ontology-based situation awareness," Information Fusion, vol. 10, no. 1, pp. 83-98, January 2009.

[19] M.R. Endsley, Theoretical underpinnings of situation awareness: a critical review, in: Situation Awareness Analysis and Measurement, Lawrence Erlbaum Associates, Mahawah, NJ, USA, 2000.

[20] W3C. RIF Core Dialect. W3C Recommendation June 22, 2010. http://www. w3. org/TR/rif-core

[21] ATLAS Transformation Language http://www.eclipse.org/gmt/atl/

[22] Casati, Roberto and Varzi, Achille, "Events", The Stanford Encyclopedia of Philosophy (Spring 2010 Edition), Edward N. Zalta (ed.), <http://plato.stanford.edu/archives/spr2010/entries/events/>.

[23] A. Scherp, T. Franz, C. Saathoff, S. Staab, F—a model of events based on the foundational ontology DOLCE+DnS ultralight, in: International Conference on Knowledge Capturing (K-CAP), Redondo Beach, CA, USA, 2009.

[24] W. van Hage, et al . Design and use of the Simple Event Model (SEM). J. Web Semantics, 2011.