NISTIR 7347

Message Validation with a Semantic Reasoning Tool

Peter Denno
Nenad Ivezic

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7347

Message Validation with a Semantic Reasoning Tool

Peter Denno
Nenad Ivezic
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

September 2006

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

Technology Administration
Robert Cresanti, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
William Jeffrey, Director

Message Validation with a Semantic Reasoning Tool

Peter Denno,
Nenad Ivezic,
National Institute of Standards and Technology,
Gaithersburg, Maryland, USA
peter.denno@nist.gov
nenad.ivezic@nist.gov

ABSTRACT

This paper describes challenges encountered in the development of a tool to validate t in OASIS UBL-based (Universal Business Language) messages. The software tool is ontology-based, using first-order logic and parts of the Suggested Upper Merged Ontology, (SUMO). The subject domain is international trade logistics. The paper reports on only the early phases of this work. It describes challenges in domain representation, and the use of semantic reasoning technology for message validation.

1.0 Introduction

Message validation concerns whether or not the messages communicated among interacting parties serve to advance their joint work. Thus if the task is to make an international purchase, then the communication between the buyer and seller ought to identify the items being purchased, the obligations of the parties, the logistics of the shipment *etc.* Among trading partners with established relationships, or where the norms of the industry are well-known, some of the information needed to perform the task may not be conveyed in the messages, but rather is built into the information systems used, and is evident as assumptions under which the persons involved operate.[1] The messages then, are not the sole force that moves the trade forward, but only the information which distinguishes this trade from others; the messages say that this particular trade concerns these particular items, and that the usual practices will be followed except as described in the details provided.

Messages refer to concepts and practices in the subject domain. The viewpoint might be peculiar to the domain and might (as is the case of international trade) range over many complex issues and interrelationships. In order to reason about the suitability of messages then, one must possess knowledge of the domain and issues involved. If the reasoning is automated, then that knowledge

1. Yet the trend in standards-based messaging is to avoid assumptions and to make intentions explicit. As much as possible, the message type defined by the standard conveys all details relevant to the task.

must be encoded somewhere in the systems performing the validation. How the information will be encoded is a principal concern in the design of the validation system. There are many possible arrangements, some of which involve encoding the information in an ontology.

This paper describes the use of an ontology for the purpose of message validation. *Section 2* considers broadly approaches to validation. *Section 3* provides an overview of a system we implemented, including our use of SUMO and the Vampire reasoner, and details of the axioms for reasoning about time. *Section 4* concludes the paper with an assessment of the approach.

2.0 Methods of Validation

In order for communication to be successful, several requirements must be met, among these: (1) the parties must agree upon a set of shared concepts to be used in the exchange; (2) statements referencing these concepts in exchanges must be encoded using a syntax agreed upon; and, (3) the messages must be conveyed by applying correctly the protocol of the messaging infrastructure. To date (2) and (3) have been the principal subject of validation efforts [16],[15].This is perhaps because messaging infrastructure and XML Schema have been active areas of development. (1) has not been the subject of nearly as much effort. This may be because the development of such tools requires substantial domain-specific knowledge, because such tools are not widely applicable, or because it is felt that the focus of such work should not be the messages, but rather characteristics of the back-end systems involved in the process. Nonetheless our work concerns (1), the set of shared concepts used in the exchange, and the validation of message content from the perspective of its consistency with an axiomatic description of the subject domain.

There are, of course, alternative mean of validation to the one chosen. The remainder of this section describes, in conceptual terms, two approaches which might be considered ends of a spectrum of possible means. One approach may be described as *criteria-directed,* the other as *constraint satisfaction*.

A *criteria-directed approach* to validation originates with a posited validation criterion, a requirement on the domain, which is in itself deemed worthy of checking. For example, we might verify that a domestic trade does not make reference to export duties to be paid. In order to implement the criteria-directed validation criteria, we simply and directly seek the answer in the data. In the case of XML-based messaging, a criteria-directed approach can be implemented using Schematron [7]. Schematron provides the structure to write rules which directly check for required values using the XML structure of the message. The Schematron approach can be characterized as not only criteria-directed but syntax-directed.

Constraint satisfaction [11] is a method by which a consistent labeling of nodes is sought in a network of node values specified by domain rules (and in our case) by the input message content. Nodes represent individuals and properties in the domain. Edges (which are bi-directional) represent relationships among individuals and properties. Domain rules are associated with sets of

edges and specify the values that may be assigned to nodes not directly specified in the data. *Figure 1* depicts the portion of a constraint satisfaction network where the buyer and seller addresses directly provide values for *buyer-country* and *seller-country* nodes. Domain rules on edges from these nodes can be used to set the value of *international-sale:* If *buyer-country* and *seller-country* have the same values, then the sale is a domestic sale, and *international-sale* is given the value *false*.

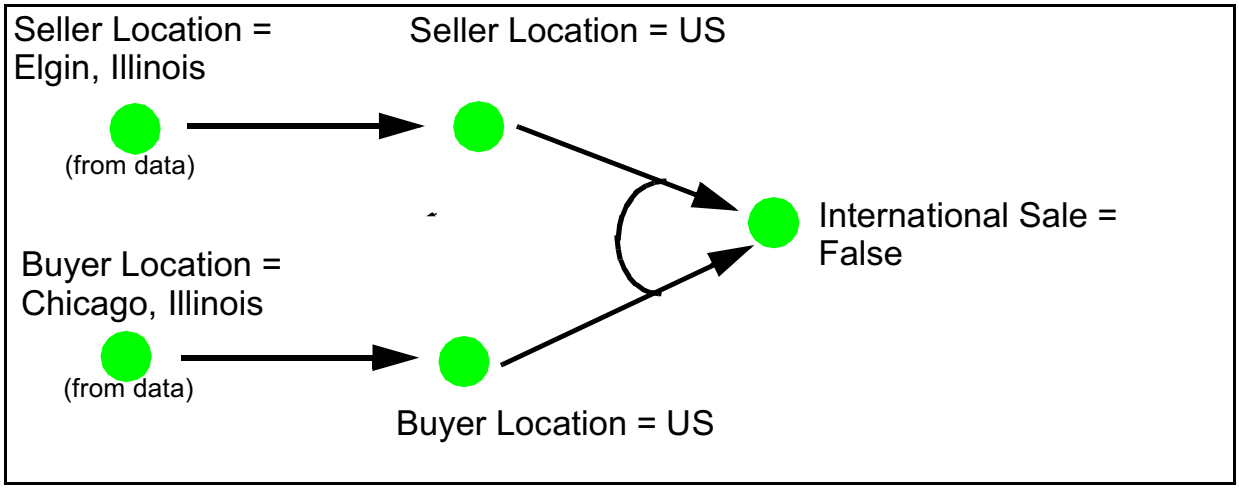

Figure 1 — Constraint propagation leading to International Sale = False

The propagation of constraints continues forward from the nodes whose values are obtained directly, from the message content, to nodes whose values are determined by the domain rules. It may be the case that two or more domain rules offer differing values for a node, or a rule allows several values. In these cases one of the values is chosen. But if that value if subsequently found to be inconsistent with other nodes in the network, the algorithm backtracks and tries another assignment to the node. If no globally consistent labeling of nodes can be found, an inconsistency in the input message content can be attributed. *Figure 2* illustrates an inconsistent network. As in *Figure 1*, *international-sale* is *false*, however *international-sale* can also be set by *incoterm-present,* a node that may get its value directly from message input referencing an Incoterm (a normative term used in international trade) [9]. Incoterms are only used in international trade.[1]This is inconsistent with the sale being domestic.

1. This may not be true soon however; there are plans to use Incoterms in domestic trade.

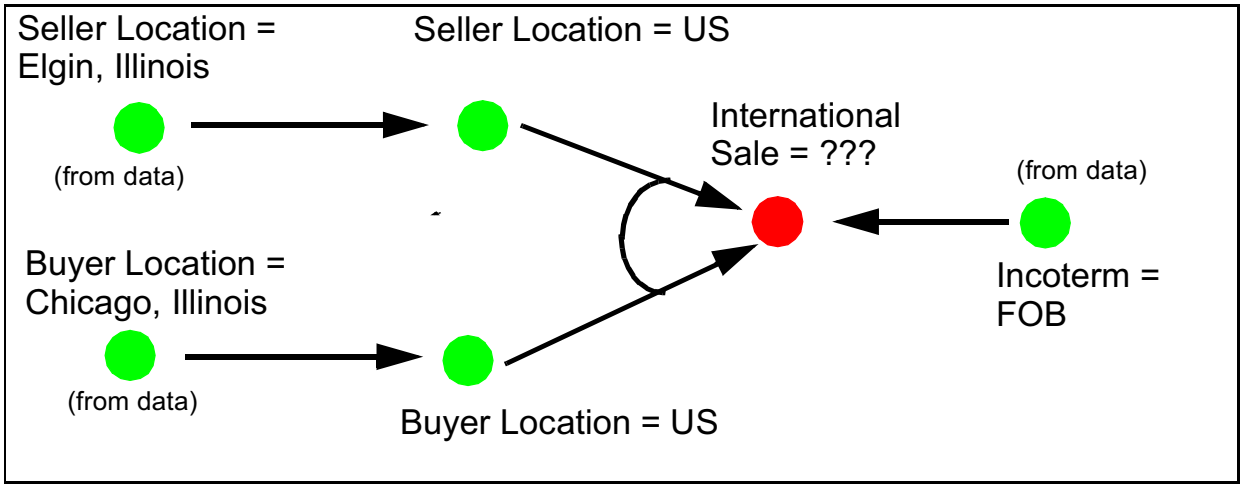

Figure 2 — Constraint propagation originating from buyer and seller locations entails domestic sale, but presence of an Incoterm entails an international sale.

The criteria-directed and constraint satisfaction methods of validation are two extremes on a spectrum of methods that might be employed. The criteria-directed method, in fact, can be viewed as a specialization of the constraint satisfaction method where the propagation of constraints is forced to run in only one direction. (See *Figure 3*).

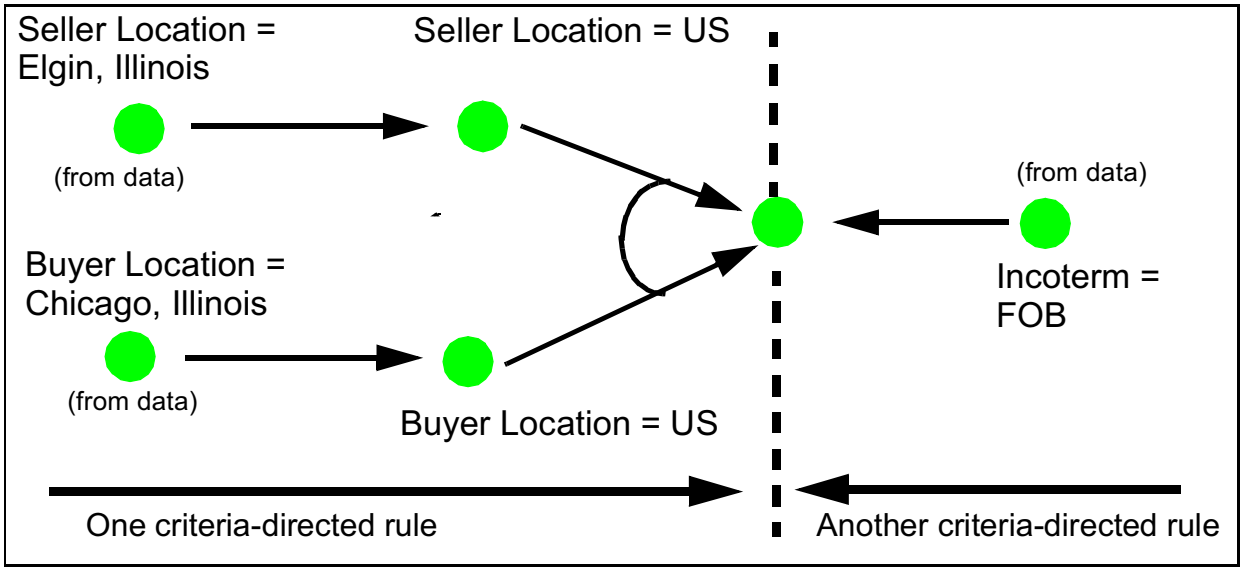

Figure 3 — In a criteria-directed method rules for each criteria are run independently.

The first-order logic approach to validation used in the trade validation project can be a middle path between the two extremes. The term *first-order* characterizes the expressivity of the language used here to represent the trade domain. In a first-order language, quantification is

allowed only over individuals. *Quantification* refers to the logical operators used to express the notions of particular and universal disposition (*e.g.* the sentence "There exists *x* such that..." and "For all *x*...") where *x* is an individual. In a higher order language, quantification might be possible over predicates and functions. (Functions can be used to identify individuals without naming them). Thus a second-order language allows sentences such as "There exists *x* that Fred hasn't told Wilma." Here *x* may be bound to a predicate (a sentence-forming relation) itself, and is existentially quantified: the sentence says that there is at least one thing (not an individual but a propsition) that Fred has not told Wilma. Higher order logical formalisms (more expressive than first-order) exist for expressing modality and propositional attitudes: believing and knowing, obligation and permission, necessity and possibility, and relative position in time. Languages less expressive than first-order include Datalog [17] (which is roughly as expressive as SQL) and description logics. Description logics [13] are an area of much recent interest due to their presumed usefulness in the semantic web. The principle inference mechanism of description logic is classification: properties attributed to an individual are used to identify the most specific type to which the individual belongs.

The potential range of considerations in the trade domain (timeliness, logistics, cost efficiency, regulations, taxes and duties, geography, risk and responsibility *etc.*) are immense and for this reason a principal concern in this work is the ability of the technology to provide a natural exposition of the domain. The most efficient means to represent the domain is with direct objective sentences unencumbered by technical concerns of the inference mechanism. A first-order language and reasoner seem well suited to these requirements. The question that arise here, though, is whether the most efficient means to represent the information is also part of the most efficient means to build a validation tool. This is a principal concern in development of the validation tool and the subject of our on-going research.

3.0 Solution Description

The principal goal of the system is to assess whether a collection of messages comprising the correspondences among trading partners are mutually consistent and consistent with a model of the trade domain embodied in the ontology. This is the sense of "validation" applied. In this section we discuss the technical details of tasks supporting the validation.

3.1 Data Extraction

The first task is, of course, to extract from the messages the domain particulars – in trade that is what is being ordered, from whom, how is it being shipped, *etc.* The messages are structured to conforms to an XML Schema specification. Presently the XML Schema used are those of UBL [20], though it should not be difficult to use similar normative XML schema, such as the Open Application Group (OAG) Business Object Documents (BODs) [18]. The system is designed not for the validation of a particular schema, but as a platform for the validation of whatever it is that might be extracted and related to the domain ontology. In fact, with a different ontology, the

system might be used to validate messages in an entirely different domain. The extraction rule editor interface is shown in *Figure 4*.

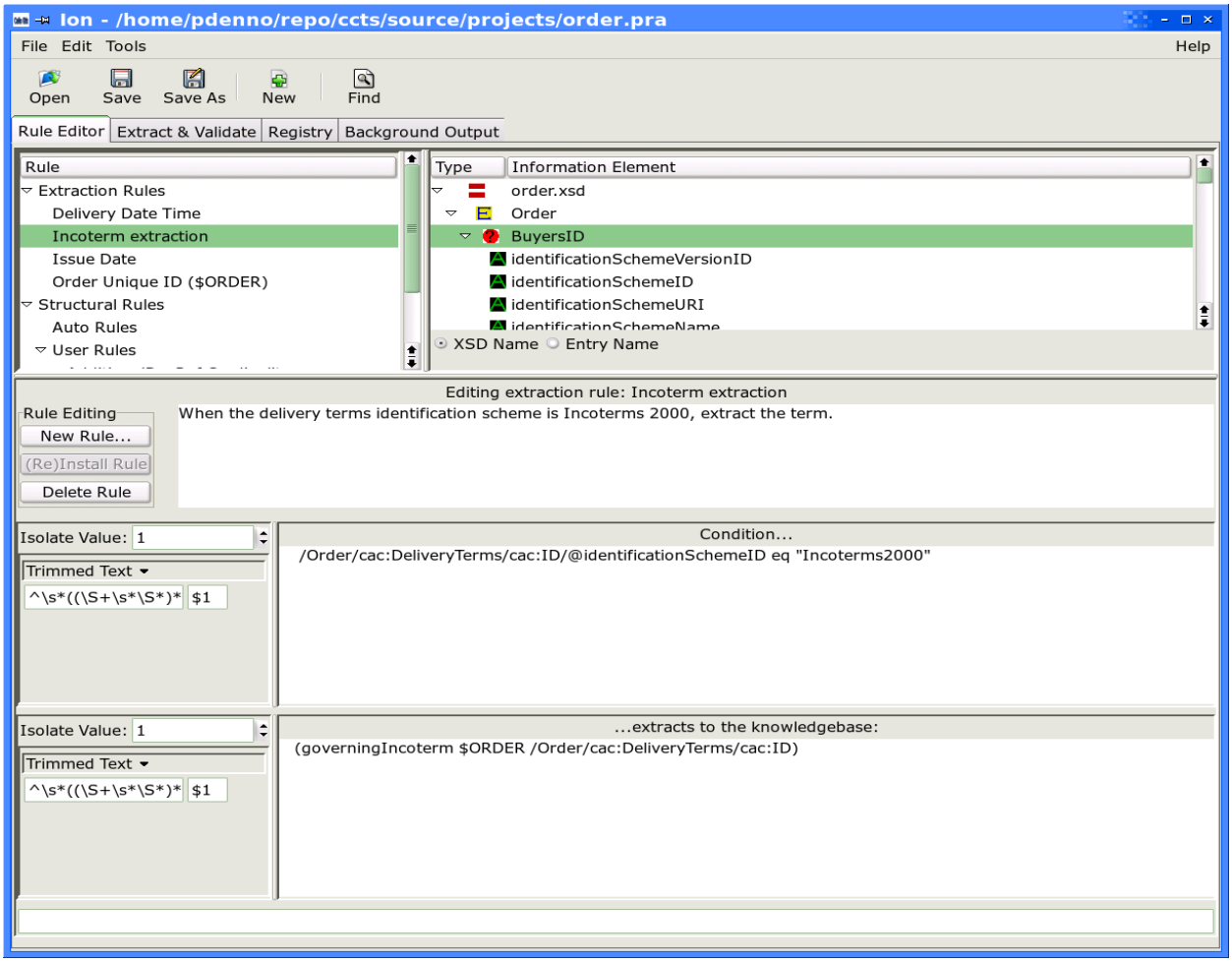

Figure 4 — The extraction rule editing interface

The data extraction process is heavily reliant on the XML structure of the messages, and on the XML Schema to which it conforms. Data extraction is implemented by *data extraction rules,* condition/action rules whose conditions are XPath [8] expressions evaluated against the message XML. The 'actions' of the rules are templates with embedded XPath. The rule action evaluates to facts that are asserted into the knowledgebase[1]. The XPath expressions are obtained by the user by referencing places in a tree describing the XML Schema (he may cut and paste from the tree to

1.Here *knowledgebase* refers to elements of the ontology that are contingent facts obtained from the messages. The fact *(Order order-n123xyz)* says that the individual *order-n123xyz* is an instance of an *Order*. *Order-n123xyz* might be derived from a string in the message, for example the order number. Unlike description logic reasoners, first-order reasoners do not concern themselves with the distinction between contingent facts and the rest of the ontology. The distinction here is to highlight the different origin of contingent facts.

the template). For example, if the editor of a rule needs to refer to the buyer's country, he navigates the tree from Order, to Buyer, to Address, *etc.*[1] *Figure 5* depicts data extraction rule editing in progress. *Figure 6* depicts the XML Schema tree for the UBL Order Schema. Note that it is often not sufficient to simply extract a string from the message document and assert it. Rather, the system may need to interpret the string as an individual of a particular data type. The left side dialog of *Figure 5* illustrates how one notes that a string should be interpreted as a date in the form 'YYYY-MM-DD'.

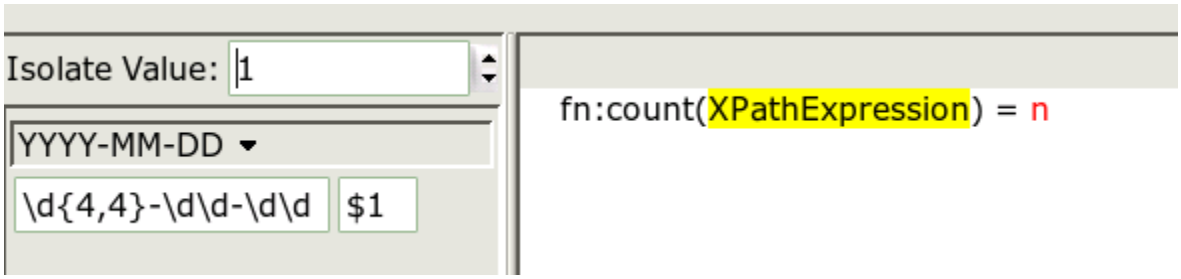

Figure 5 — Highlighted text identifies a target for a XPath Expression that may be referenced from the schema of the message type, depicted in Figure 6.

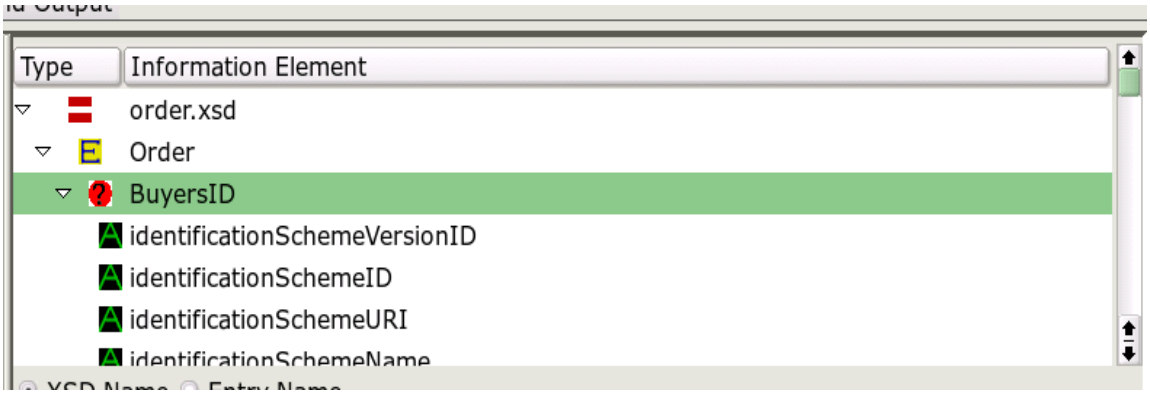

Figure 6 — The XML Schema is analyzed to produce a tree which is referenced in the definition of extraction rules.

When the data extraction rules are executed against messages they assert contingent facts into the knowledgebase. It is then possible to run the inference mechanism to assess the validity of the messages. Note that is also possible to write rules whose actions directly check for the presence of certain information in a message. However, this ability is not well-developed in the system since that functionality is already available in tools such as Schematron.

There are then three types of "knowledge engineer" who might use the system: (1) those who can

1.Note that the tree represents the XML Schema of the message and not the Document Object Model of a particular message. Thus rules can be defined for all messages corresponding to the XML Schema.

write single-message Schematron like rules, (2) those who can write data extraction rules, and must therefore be familiar with the domain ontology, and; (3) those who can write the domain ontology. This final group must be expert modelers in the Knowledge Interchange Format (KIF) and the Suggested Upper Merged Ontology, (SUMO) [1] and they must have access to domain expertise.

In our early design conception of the system, the intention was that data extraction would be aided by the use of UBL Core Components.[19] The idea being that knowing the Core Component Type (CCT) of data in the messages would help to identify the corresponding concept or relation in the ontology. However, our investigation revealed that the types defined by the CCTs were not sufficiently distinguished (neither the definition, nor in practice, their use) to make this possible. In particular, reading the CCT specification [19], we could not grasp the intended distinction between the code and identifier types. In retrospect, the UBL Business Information Entities (BIEs) [20] may have been the more appropriately contextualized information to correlate to the ontology concepts and relations. This may be investigated further.

3.2 Inference

The system uses an open source version of the Vampire [5] first-order reasoner to validate message content against the trade ontology. Vampire allows significant control over the inference process. What is described here only touches on a few of its aspects.

The principal form of reasoning that is employed by Vampire is *ordered resolution with equality*. *Ordered* here refers to any technique used to define a precedence on terms. The ordering is used particularly in expressions involving equality, to judge which of two terms is "simpler," thus it provides direction in the search for a solution. *Resolution* refers to a rule of inference whereby clauses with terms that are similar, but one being the negation of the other, can be combined into a clause which eliminates those terms. Most first-order reasoners use resolution as one of their means of inference.

Resolution and other rules of inference used in first-order reasoners require formulas in a syntactically simpler (than, say, KIF[1]) form called *clause normal form.* Although Vampire transforms ("clausifies") the input transparently, clausification is a significant issue in this project because some parts of the proof provided by the reasoner may remain in clausal form, not the original KIF.[1]

Our use of Vampire uses *proof by refutation.* In this method, the formula that you are trying to prove (*e.g.* a sentence assessing some aspect of the trade) is negated and added to the input to the reasoner. With this input the reasoner searches for a contradiction using a method called *saturation* – basically, a broad search. If a contradiction is found, then the original sentence must be true, since its negation is not consistent with the ontology. Proof by refutation using ordered resolution is complete, meaning that if a contradiction exists, it will eventually be found. Problems posed to Vampire specify a time-out value that tells Vampire when to stop looking for a

1.Vampire accepts several forms of input. This project uses its SUO-KIF interface, which is the language used in SUMO.

solution. The time-out value is also used dynamically in the management of the search.

An advantage of proof by refutation is that if an inconsistency exists anywhere in the ontology, it will be found : when an inconsistency is present, any statement can be shown to be true and the proof will include reference to the inconsistent statements. This characteristic is useful in ensuring that the ontology is constructed accurately.

3.3 Ontology

The messages communicated among trading partners make reference to a large span of concepts and relationships from the world of trade and logistics. In contrast to criteria-directed validation, an ontology-based approach seeks first to describe the concepts, relationships, and constraints in that world. The inference mechanism then, by way of its proof theory and the proofs sought for various concerns[1], identifies inconsistencies arising from the contingent facts derived from the messages. The knowledgebase is "inconsistent" in the sense that some contingent facts contradict constraints described by the ontology – constraints that reflect the real-world.

Most informal domains[2] rest upon a collection of concepts and constraints that concern not the minutia of the subject matter, but "human consensus reality" – the multitude of observations that underlay the common elements of our understanding of the world. This foundation is called an *upper ontology*[2], [3], [1]. It remains to be proven whether or not a single upper ontology can provide the foundation for various, dissimilar domains; this has been the subject of much debate over the last decade. However, our experience with this project suggests that the idea of a multi-purpose upper ontology has merit. We built our (currently small) trade ontology on top of an upper ontology containing about two thirds of SUMO.[3] We believe that the quality of our work is improved by the use of SUMO.

SUMO (The Suggested Upper Merged Ontology) is the on-going work of an IEEE standardization effort. SUMO's 4500 axioms are written in SUO-KIF (Standard Upper Ontology - Knowledge Interchange Format) [21], a dialect of Common Logic [6].The structure of SUMO is described in *Figure 7*, from the SUMO documentation [1].

1."The proofs sought for various concerns" suggests that our ontology-based approach is criteria-directed. Indeed, our experimentation to date has been criteria-directed, but in theory it should be possible to identify inconsistencies in the knowledgebase by posing very general problems to the reasoner – much more like constraint satisfaction than criteria-directed validation.

2.By *informal domain* we mean the great number of domains, trade, wine tasting, or real estate *etc*. that cannot be described with mathematical precision.

3.About one third of SUMO was clearly not relevant to our task and was removed to improve the performance of the reasoner. We wrote software tools to ensure that the transitive closure of predicate symbols referenced was fully within our subset. These tools also ensured the "structural integrity" of the ontology (*e.g*. that everything is an instance of something, and that every class hierarchy terminates in Entity, *etc.*).

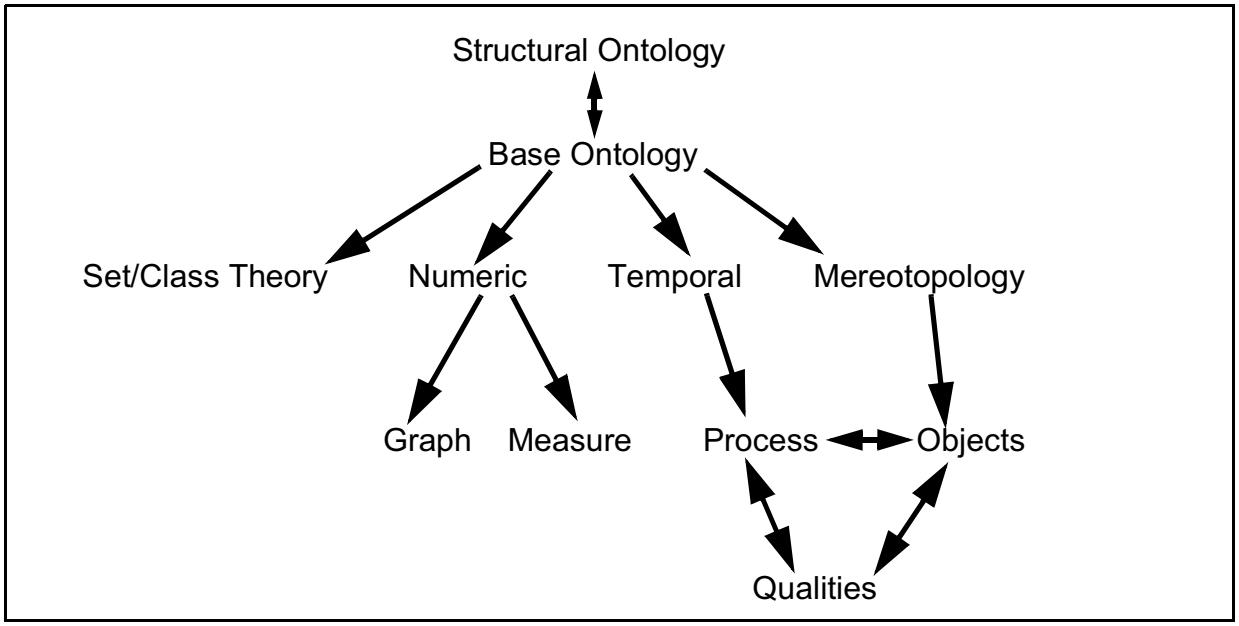

Figure 7 — The organization of the Suggested Upper Ontology (SUMO)

With the exception of the usual five logical connectives and the quantification operators, the "modeling language" of SUMO is defined in SUMO. Hence there is very little outside of SUO-KIF and SUMO itself that needs to be understood to read the ontology. This is due in part to the fact that in SUO-KIF, predicate symbols can be the subject of predication. For example, the partition of a class is a common notion in information modeling. partition is defined in *Figure 8*.

```
(instance partition Predicate)
(instance partition VariableArityRelation)
(domain partition 1 Class)
(documentation partition "A &%partition of a class C is a set of
mutually &%disjoint classes (a subclass partition) which covers C.
Every instance of C is an instance of exactly one of the subclasses
in the partition.")

(<=>
   (partition @ROW)
   (and (exhaustiveDecomposition @ROW)
        (disjointDecomposition @ROW)))
```

Figure 8 — SUMO "information modeling primitives" are defined in the ontology itself.

As illustrated in *Figure 8*, partition is the subject of predication by the instance predicate, a predicate whose first domain is any Entity, (everything is an Entity) and whose second domain is a SetOrClass. The symbol @ROW denotes a *sequence variable*[1] (ordinary variables in KIF begin with '?'). A sequence variable is a means to reference collectively, as a sequence, an arbitrary

number of positions in a relation. It is particularly useful in binding the roles of a variadic relation such as partition, exhaustiveDecomposition, and disjointDecomposition. The axioms in *Figure 8* defines partition to be a variable arity relation for which the exhaustiveDecomposition and disjointDecomposition relations hold. Thus partition is just syntactic sugar for other relations. Using this predicate, SUMO may contain formula such as (partition Animal Vertebrate Invertebrate), instead of a more complex axiom referencing disjointedness and exhaustive decomposition explicitly.

The notion of a meta-model of a language is prevalent in some information modeling languages. A concept can be both positioned in a domain and it can be noted that the concept is an abstraction. These are independent observations. With technologies such as UML we would make these two observations at different "meta-levels." The latter would be in the meta-model of the language and would not be accessible to tools that only possessed the domain model. In SUMO there is no such separation. For example, Physical is a subclass of Entity: (subclass Physical Entity) but it is an instance of Abstract: (instance Physical Abstract). These two assertions correspond to the two viewpoints of the domain and language meta-model respectively. In a first-order language (in contrast to a description logic) it is possible for the same individual to participate in an instance relationship, and a subclass relationship. This property of a first-order language allows one to relate an individual in the domain to a "schema" that might be used to characterize it.

3.4 Reasoning about Time

Reasoning about trade logistics often entails reasoning about time (*e.g.* the period of time a contract is in force, due dates, expirations, time in transit, *etc*.). In this work we used, with some modification, the Process Specification Language (PSL) duration theory [12] for reasoning about time.

There are two fundamental types essential to reasoning in this domain: timepoints and time durations. *Timepoints* are stateless individuals on which there is a complete ordering. A time duration is just a quantity of something (time). The magnitude of the quantity must ultimately be related to a metric based on ordered timepoints.

To reason about a domain notion of dates, (e.g. 8PM, Aug 23, 2005) it is necessary to define a mapping from the timepoint individuals to dates. We chose a countable, non-dense representation of a timeline where the successor of a timepoint is 1 minute in duration after the given timepoint. We defined a special "origin" timepoint. T0, that maps to the date 00:00:00, Jan 1, 2000. The notion of an origin is not defined in PSL, yet it seems necessary if the timepoint individuals are to be related to a domain notion of dates. Some notions in the PSL duration theory were not relevant to the domain: *Axiom 5,* the notion of an additive inverse of time durations, (which implies something like "negative durations"); *Axiom 19*, the notion that the additive inverse of the value

1.Sequence variables are not part of a first-order language. We wrote a preprocessor to expands sequence variables to multiple formulas, one with a single regular variable, one with two, *etc*. Up to an arbitrary seven variables. This is done before sending the input to Vampire for clausification.

of the duration function on (T1,T2) is the value of the duration function on (T2,T1), and; *Axiom 15* the notion that adding any duration other than max+ to max- is max- . (max- is a constant that corresponds to the notion of an "infinite negative duration.")

The axioms (not part of PSL) that concern the successor of a timepoint are provided in *Figure 9*.

```
(instance TimePointSuccessorFn UnaryFunction)
(instance TimePointSuccessorFn TotalValuedRelation)
(domain TimePointSuccessorFn 1 TimePoint)
(range TimePointSuccessorFn TimePoint)
 ;;; successor of a Timepoint is 1 minute later.
(<=>
   (equal ?SUCC (TimePointSuccessorFn ?T))
   (and (duration (TimeIntervalFn T0 ?T) ?DUR)
        (duration (TimeIntervalFn T0 ?SUCC) ?DUR1)
        (equal ?DUR1 (DurationAddFn ?DUR (MeasureFn 1 MinuteDuration)))))
;;; The successor of a TimePoint is unique.
(=>   (equal (TimePointSuccessorFn ?T1)
             (TimePointSuccessorFn ?T2))
   (equal ?T1 ?T2))
;;; Every TimePoint but T0 is the successor of some TimePoint.
(=>
 (and
   (instance ?T TimePoint)
   (not (equal ?T T0)))
 (exists (?T1)
   (equal ?T (TimePointSuccessorFn ?T1))))
;;; There is no TimePoint between ?T and (TimePointSuccessorFn ?T)
(=>
 (and (instance ?T1 TimePoint)
        (instance ?T2 TimePoint)
        (instance ?T3 TimePoint)
        (equal (TimePointSuccessorFn ?T1) ?T2)
        (not (equal ?T3 ?T1))
        (not (equal ?T3 ?T2)))
  (or
    (earlier ?T3 ?T1)
    (earlier ?T2 ?T3)))
```

Figure 9 — Timepoints are ordered. The successor occurs 1 minute later. There are no timepoints between a timepoint and its successor.

A key task that the reasoner should be capable of performing is that of recognizing the equivalence between a time interval described by two timepoints and a time interval described by the first timepoint and a duration. For example, the interval starting at timepoint 2 and ending at timepoint 6 is equivalent to the interval starting at timepoint 2 and running a duration of 4.[1] The

1.For ease of exposition here we use integers for timepoints and durations. The implementation uses timepoint individuals (e.g. (instance T2 Timepoint), (duration (TimeIntervalFn T0 T2) (MeasureFn 2 Minute)). The implementation uses durations individuals that are units of measure (e.g. (MeasureFn 2 Minute)).

principal observation required to make the inference is that timepoints and durations are related through a notion of addition. PSL defines such a function, time_add, but does not relate it to Peano arithmetic (presumably other theories of arithmetic might be relevant to one's domain). *Figure 10* describes Peano arithmetic on time intervals, and the relation to the DurationAddFn.

```
(=>
 (and (instance ?T TimePoint)
      (duration (TimeIntervalFn T0 1) ?D1)
      (duration (TimeIntervalFn T0 ?T) ?D2)
      (duration (TimeIntervalFn T0 (TimePointSuccessorFn ?T)) ?D3))
 (equal (DurationAddFn ?D1 ?D2) ?D3))

(=>
 (and
   (instance ?T1 TimePoint)
   (instance ?T2 TimePoint)
   (duration (TimeIntervalFn T0 1) ?D)
   (duration (TimeIntervalFn T0 ?T1) ?D1)
   (duration (TimeIntervalFn T0 ?T2) ?D2)
   (=>
    (exists (?TP2)
       (duration (TimeIntervalFn T0 ?TP2) ?D3)))
    (equal
     (DurationAddFn ?D1 ?D2)
     (DurationAddFn ?D1 (DurationAddFn ?D ?D3))))
```

Figure 10 — Recursive definition of DurationAddFn

If it were not the case that the reasoner had a built-in ability to do arithmetic, it would be necessary to provide the above axioms to the reasoner. Of course, arithmetic reasoning using this approach would be unacceptably slow. Fortunately, Vampire has the ability to do arithmetic. The only thing necessary then, is to define DurationAddFn such that it uses units of measure consistently. The axiom is depicted in *Figure 11*.

```
(<=> (equal ?T2 (DurationAddFn ?T1 ?D))
     (exists (?UNIT ?TD ?N1 ?N2)
        (and
          (duration (TimeIntervalFn T0 ?T1) (MeasureFn ?N1 ?UNIT))
          (duration (TimeIntervalFn T0 ?TD) ?D)
          (equal (MeasureFn ?N2 ?UNIT) ?D)
          (duration (TimeIntervalFn T0 ?T2)
                    (MeasureFn (AdditionFn ?N1 ?N2) ?UNIT)))))
```

Figure 11 — DurationAddFn uses Vampire's arithmetic functions.

3.5 Units of Measure

As the above axioms suggest, SUMO has axioms to describe physical quantity. There are two subclasses of PhysicalQuantity: (1) instances of subclass ConstantQuantity are represented by the magnitude (a RealNumber) and UnitOfMeasure arguments to MeasureFn such as illustrated by (MeasureFn 1 MinuteDuration) in *Figure 9* above; (2) instances of subclass FunctionQuantity are compositions of ConstantQuantity that are the value of certain functions. For example, (DensityFn (MeasureFn 2 Gram) (MeasureFn 1 Liter)) represents 2 grams per liter.

Subclasses of UnitOfMeasure include the Systeme International base and derived units, and units for quantities that are not physical, *e.g.* UnitedStatesDollar. We found the SUMO approach to units of measure to be quite sufficient for our needs.

3.6 Buyer, Seller, and Sales Contract

SUMO defines notions of buying and selling. Buying and Selling are subclasses of FinancialTransaction, Transaction, ChangeOfPossession, and finally Process. SUMO typically does not reify the agents of processes as types, but for convenience we defined buyer and seller as illustrated in *Figure 12.*

```
(instance seller CaseRole)
(domain seller 1 Selling)
(domain seller 2 CommercialAgent)
(documentation seller "(seller ?SELLING ?AGENT) means that
?AGENT participates in a Selling process ?SELLING".)

(instance buyer CaseRole)
(domain buyer 1 Buying)
(domain buyer 2 CognitiveAgent)
(documentation buyer "(buyer ?BUYING ?AGENT) means that
?AGENT participates in a Buying process ?BUYING".)

;;; Money goes from the buyer to seller.
(=>
  (and
   (buyer ?BUYING ?BUYER)
   (seller ?SELLING ?SELLER)
   (subProcess ?BUYING ?PURCHASE)
   (subProcess ?SELLING ?PURCHASE)
   (instance ?PURCHASE FinancialTransaction)
   (patient ?PURCHASE ?OBJECT)
   (monetaryValue ?OBJECT ?MONEY))
 (exists (?PAYMENT)
     (and
        (subProcess ?PAYMENT ?PURCHASE)
        (instance ?PAYMENT Payment)
        (transactionAmout ?PAYMENT ?MONEY)
        (destination ?PAYMENT ?SELLER)
        (origin ?PAYMENT ?PUYER))))
```

Figure 12 — Axioms on buyer and seller

A useful characteristic of SUO-KIF is its formula backquoting ability, which allows one to bind a formula to a quantified variable. Though the result resembles a higher order sentence, it is not treated as one -- the quoted formula is essentially treated as a opaque structure and ignored by the reasoner. One use of the backquote might be to define the details of a sales contract and assert that it contains obligations. Our system stores all KIF formulas in a discrimination tree [22], allowing us to query for a backquoted formula. This might be used, for example, to query the parts of a contract, and with that information, perform specific validation tasks.

Figure 13, illustrates how it can be asserted that the embedded contract proposition contains an obligation.

```
;;; All purchases involve a "sales contract" which contains Obligations
(=>
 (and
  (buyer ?BUYING ?BUYER)
  (seller ?SELLING ?SELLER)
  (subProcess ?BUYING ?PURCHASE)
  (instance ?PURCHASE FinancialTransaction))
 (exists (?COMMIT ?SENTENCE ?PROPOSITION)
    (and
      (instance ?COMMIT Committing)
      (property ?COMMIT Contract)
      (agreementMember ?COMMIT ?BUYER)
      (agreementMember ?COMMIT ?SELLER)
      (instance ?SENTENCE Sentence)
      (represents ?SENTENCE ?PURCHASE)
      (containsInformation ?SENTENCE ?PROPOSITION)
      (property ?PROPOSITION Obligation))))
```

Figure 13 — Use of predicate containsInformation to bind a proposition, which is an obligation

3.7 Codes

Industry uses many normative codes in its communications. By *code* we mean the normative definition of the individuals of some type, and the association of a text identifier with each individual. Incoterms 2000 [9] and United Nations LOCODE [14] are examples of codes. The individual "FOB" in Incoterms refers to a manner of shipping, also known as "Free on Board," where responsibilities, costs, and risks transfer from the seller to the buyer as the goods are loaded aboard the ship used in its main carriage. The individual "US GAI" in LOCODE refers to Gaithersburg, Maryland, USA.

There can be no uniform treatment of codes in the ontology because there is no uniformity across the various codes. For example, there are currently about 50,000 LOCODE individuals defined, and each individual specifies a small set of simple properties such as whether or not the location has a port, an airport, a rail terminal, a road terminal, and its longitude and latitude. On the other hand, there are only 13 Incoterm individuals and each refers to a complex collection of concerns that may be legally binding.

The challenge that a large code such as LOCODE presents is simply the matter of how to get the parts that are needed into the reasoner when they are needed. Database lookup is sufficient. The challenge that a complex code such as Incoterms presents is the matter of how the viewpoint and terminology of the specification should be integrated into the ontology: As specification terminology is replaced with ontology terminology, verifying that the code individuals have been correctly represented becomes increasingly difficult. As specification terminology is allowed to dominate, it becomes increasingly difficult to make use of the fundamental inferences provided by the upper ontology. The solution, of course, is just to be careful; study the axioms of the upper

ontology to be sure that they support just the intended meaning. The larger issues at play here are (1) whether or not the upper ontology can deliver improved efficiency compared to an implementation that does not use one, and; (2) how far the upper ontology can venture into detail before it begins to present a viewpoint that conflicts with that of the subject domain. With respect to (1) we are fairly certain that the cost of understanding the details of the upper ontology are less than the cost of trying to specify the same detail *ab initio*. With respect to (2), our experience is that SUMO, at least, has struck about the right balance.[1]

4.0 Conclusion

A principal goal of this project was to gauge the efficiency of using a first-order reasoning tool in message validation tasks. We found that many of the challenges involved with the implementation (data extraction, and presentation of results, for example) were just as they would be with conventional technology. The expressivity of a first-order language, and the "running start" we gained by using SUMO were definite advantages. A first order language seems well suited for expressing complex notions in the trade domain. However, there are still problems to resolve and more design terrain to explore. We have not yet considered how to pose broad validation task to Vampire; we have only attempted focused queries. The proof returned from a broad query will have to be analyzed and presented to the user. The complexity of this has not yet been explored. These task might be accomplished easier had we used a description logic, but we may trade expressivity in doing so.

The trade ontology is currently very small and the authors are not experts in the domain. We hope to participate in an upcoming Automotive Industry Action Group (AIAG) initiative in international supply chain logistics, [10] where we might gain the help of domain experts.

5.0 References

[1] Niles, I., and Pease, A., *Towards a Standard Upper Ontology*. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Chris Welty and Barry Smith, editors, Ogunquit, Maine, October 17-19, 2001.

[2] OpenCyc, *OpenCyc Site on SourceForge*, http://sourceforge.net/projects/opencyc/, 2005.

[3] Maolo, C. et al., *WonderWeb Delivery D17, The WonderWeb Library of Foundational Ontologies, Preliminary Report*, May 29, 2003.

[4] Robinson, J., A., and Voronkov, A., (editors) *Handbook of Automated Reasoning,* MIT Press, 2001.

1.There is however a "mid-level" ontology associated with SUMO, called MILO, which contained some axioms about financial concepts that we had hoped to make use of. However, we did not find the match to our needs to be very good.

[5] Raizanov, A., *Implementing an Efficient Theorem Prover*, PhD Dissertation, University of Manchester, UK, July 2003.

[6] ISO/IEC, *Information Technology – Common Logic (CL) – A Framework for a Family of Logic-based Languages*, JTC 1 / SC32, N1330, http://phlebus.tamu.edu/cl/ June 23, 2005.

[7] ISO/IETC, *Information Technology – Document Schema Definition Language (DSDL) – Part 3: Rule-based validation – Schematron*, 19757-3 Schematron, 19757-3, JTC 1 / SC34, Final Committee Draft, 2005.

[8] W3C, *XML Path Language (XPath) 2.0*, W3C Working Draft, http:/www.w3.org/TR/ xpath20/ April 4, 2005.

[9] Reynolds, F., *Incoterms For Americans*, International Projects, Inc., 2001.

[10] AIAG *Material Flow Initiatives – Strengthen Off-Shore Supply Chains, Town Hall Meeting*, Troy, Michigan, http://www.aiag.org/events/material_flow.cfm, June 30, 2005.

[11] Tsang, E. P. K., *Foundations of Constraint Satisfaction*, Academic Press, 1994.

[12] ISO, *Industrial automation systems and integration – Process specification language – Part 13: Outer Core, Duration Theory*, http://www.mel.nist.gov/psl/psl-ontology/part13/duration.th.html, 2004.

[13] Baader, F., Nutt, W., *Basic Description Logic*, http://www.inf.unibz.it/~franconi/dl/course/ dlhb/dhlb-02.pdf, 2002.

[14] UNECE, *United Nations Codes for Trade and Transportation Locations, UN/LOCODE*, United Nations Economic Commission for Europe, http://www.unece.org/cefact/locodes/service/ main.htm, 2005

[15] NIST, *NIST Manufacturing Business to Business (B2B) Interoperability Testbed*, http:// www.mel.nist.gov/msid/b2btestbed/, 2005.

[16] Kulvatunyou, B., Ivezic, N., Martin, M., Jones, A.T., *A Business to Business Interoperability Testbed: An Overview*, The Fifth International Conference on Electronic Commerce, Pittsburg, PA. October, 2003.

[17] Dbiteboul, S., Hull, R., Vianu, V., *Foundations of Databases: The Logical Level,* Addison-Wesley, 1994.

[18] Open Application Group, *OAGIS 9.0 Schemas*, http://www.openapplications.org/downloads/ oagisdownload.htm.

[19] UN/CEFACT, *ebXML Core Components Technical Specification, Version 1.85*, United Nations Centre for Trade Facilitation and Electronic Business, September 30, 2002.

[20] OASIS, *Universal Business Language 1.0*, Organization for the Advancement of Structured Information Standards (OASIS), September 15, 2004.

[21] IEEE, *IEEE Standard Upper Ontology (SUO) Study Group: Knowledge Interchange Format*, http://suo.ieee.org/SUO/KIF/, December, 2003.

[22] Norvig, P., *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*, Morgan Kaufmann Publishers, 1992.