2010

An Ontology-Based Assistant For Analyzing Agents' Activities

Roland Johnson
North Carolina Agricultural and Technical State University

AN ONTOLOGY-BASED ASSISTANT FOR ANALYZING AGENTS' ACTIVITIES

by

Roland Johnson

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Department: Computer Science
Major: Computer Science
Major Professor: Dr. Albert Esterline

North Carolina A&T State University
Greensboro, North Carolina
2010

School of Graduate Studies
North Carolina Agricultural and Technical State University

This is to certify that the Master's Thesis of

Roland Johnson

has met the thesis requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2010

Approved by:

_____
Dr. Albert Esterline
Major Professor

_____
Dr. Gerry Dozier
Department Chairperson

_____
Dr. Alan Letton
Interim Associate Vice Chancellor for
Research and Dean of Graduate Studies

DEDICATION

I dedicate this thesis to my daughter, Destiny. Your life has brought me unexplainable love and joy. I love you with all my heart.

BIOGRAPHICAL SKETCH

Roland Johnson was born on September 5, 1968, in Martinsville, Virginia. He received a Diploma in Electrical/Electronics Engineering Technology from Danville Community College in 1998 and a Bachelor of Science degree in Computer Science from North Carolina A&T State University in 1996.

ACKNOWLEDGMENTS

I would like to thank Dr. Albert Esterline for his many years of direction, assistance, guidance, and understanding. Special thanks should be given to my wife, Anishia Johnson, for her encouragement and assistance through some very difficult times in my life. I love you, sweetheart.

TABLE OF CONTENTS

LIST OF FIGURES viii

ABSTRACT ix

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 BACKGROUND 6

2.1 Representing Objects and Qualitative Spatial Relations 6

2.2 Linguistic Notions and FrameNet 8

2.2.1 Frame Semantics 11

2.2.2 Example Lexical Unit 16

2.2.2.1 Move Lexical Unit 16

2.2.2.2 Survey Lexical Unit 18

2.3 Jena 19

2.4 Google Earth and KML 21

2.4.1 Google Earth 21

2.4.2 KML 23

2.4.3 Google Earth API 26

2.5 XSLT 27

CHAPTER 3 SOFTWARE FOR ANALYSTS AND MODELERS 31

3.1 Initial KML Files 32

3.2 The Analyst 33
3.3 The Modeler 37
CHAPTER 4 COGNITIVE IMPLICATIONS 40
4.1 Intentions and Possibilities Open to the Player 40
4.2 Situation Awareness 43
CHAPTER 5 CONCLUSION 45
BIBLIOGRAPHY 49
APPENDIX A USING JENA WITH RDF 53
APPENDIX B SPARQL/JENA 56
APPENDIX C PROCESSING OWL WITH JENA 60

LIST OF FIGURES

FIGURE **PAGE**

1 The Theme Moving along a Path from the Source to the Goal 3

2 Graphical Representation of an RDF Resource 19

3 Graphical Representation of an RDF Triple 20

ABSTRACT

Johnson, Roland. AN ONTOLOGY-BASED ASSISTANT FOR ANALYZING AGENTS' ACTIVITIES. **(Advisor: Albert Esterline)**, North Carolina Agricultural and Technical State University

This thesis reports on work in progress on software that helps an analyst identify and analyze activities of actors (such as vehicles) in an intelligence-relevant scenario. A system is being developed to aid intelligence analysts, IAGOA (Intelligence Analyst's Geospatial and Ontological Assistant). Analysis may be accomplished by retrieving simulated satellite data of ground vehicles and interacting with software modules that allow the analyst to conjecture the activities in which the actor is engaged along with the (largely geospatial and temporal) features of the area of operation relevant to the natures of those activities. Activities are conceptualized by ontologies. The research relies on natural language components (semantic frames) gathered from the FrameNet lexical database, which captures the semantics of lexical items with an ontology using OWL. The software has two components, one for the analyst, and one for a modeler who produces HTML and parameterized KML documents used by the analyst. The most significant input to the modeler software is the FrameNet OWL file, and the interface for the analyst and, to some extent, the modeler is provided by the Google Earth API.

CHAPTER 1

INTRODUCTION

This thesis reports on research in progress that identifies and analyzes activities of actors (such as vehicles) in an intelligence-relevant scenario. A system is being developed to aid intelligence analysts, IAGOA (Intelligence Analyst's Geospatial and Ontological Assistant) [1]. Analysis may be accomplished by retrieving simulated satellite data of ground vehicles and interacting with software modules that allow the analyst to conjecture the activities in which the actor is engaged along with the (largely geospatial and temporal) features of the area of operation relevant to the natures of those activities. Activities, such as movements of the vehicles, are conceptualized by ontologies. An ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain and may be used to define the domain. The research relies on natural language components (semantic frames) gathered from a large lexical resource namely, FrameNet [2], which is based on Fillmore's frame semantics. FrameNet captures the semantics of lexical items with an ontology using the Web Ontology Language (OWL). It provides the ability to define activities by the targeted actors. Examples of such activities are *move, hide,* and *survey*. IAGOA associates an intelligence analyst's understanding of a player's activities with the geospatial features of the area of operation in which these activities take place. (The neutral term "player" refers to the monitored agent, which could be a

person but also, e.g., an unmanned vehicle. We use masculine pronouns simply for ease.) It will also help an analyst articulate conjectures about the player's activities, stated in terms of verbs. The software has two components, one for the analyst, and one for a modeler who produces HTML and parameterized KML documents used by the analyst. The most significant input to the modeler software is the FrameNet OWL file, and the interface for the analyst and, to some extent, the modeler is provided by the Google Earth API. The Jena framework for Semantic Web applications is used extensively for processing the FrameNet OWL file, and XSLT is used for editing KML documents.

As an example, Figure 1 shows a hiker (the "Theme") moving along a path from the source, where he started, to the goal, to which the analyst conjectures he is headed. The map already highlights geospatial features that can be significant for human activities. These features are in the nature of Gibson's affordances [3], which are clues in the environment that indicate possibilities for action, but here the clues are on the map. We have clues of both what can be done (cf. the tracks) and what cannot be done (cf. the peaks and streams), and we see what might be a desirable place to be (cf. the building). The features have different dimensionalities. We think of the tracks and streams as one-dimensional since we are not particularly interested in their widths except insofar as they impede or facilitate movement, and this information could be attached to them as attributes. Where tracks and streams terminate or meet are effectively zero-dimensional landmarks as are the building and where a track terminates.

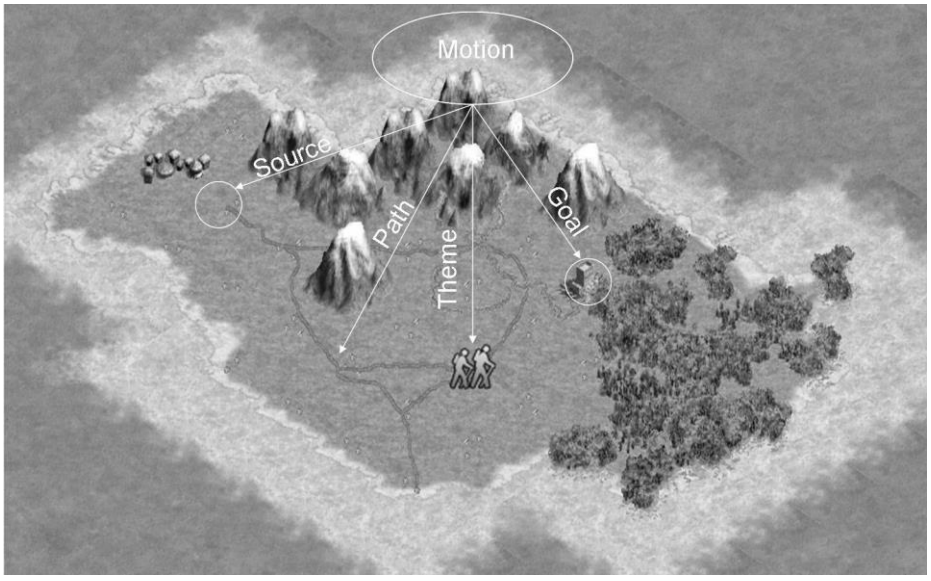

Figure 1. The Theme Moving along a Path from the Source to the Goal

Actual activities are expressed in the first instance by verbs, but a verb by itself does not have total meaning until it is placed into context with its "frame elements" (or FEs), which frame semantics characterizes as roles filled by the denotations of the complements of the verb. For example, *move* evokes the *Motion* frame, which characteristically includes the roles *Theme* (the thing moving), *Source* (where it starts), *Path* (along which it moves), and *Goal* (where it ends) as seen in Figure 1. In the current domain, the roles are often expected to be filled with geospatial features. The case shown in Figure 1 involves understanding not only the (im)possibilities and attractions or aversions afforded by the geospatial figures and what it means to move but also how the *Motion* frame is instantiated in the depicted area of operation. In this setting, the analyst has observed the player starting his motion at the track terminus shown, which fills the *Source* role, and has observed him moving along the track to the location shown. The

building is the obvious thing of interest, so a good guess has it fill the *Goal* role. We already know part of the path, and the obvious thing for our hiker heading for the building to do is to continue on this track to its intersection with the stream and then follow it downstream to the building. Concatenating these segments together gives a good guess for what fills the *Path* role. Some roles, however, are temporal (for example, *Motion* includes *Speed* or *Duration*) and some are non-spatiotemporal, such as *Manner* (e.g., stealthily).

Now, activities are expressed in the first instance by verbs, but a verb by itself does not have total meaning until placed into context with its "frame elements" (or FEs), which are roles filled by the denotations of the verb's complements. For example, *move* evokes the *Motion* frame, which characteristically includes the roles *Theme* (the thing moving), *Source* (where it starts), *Path* (along which it moves), and *Goal* (where it ends). In IAGOA, the analyst identifies the fillers for the roles and the system simulates the player and updates the display showing what really transpires. If the conjecture deviates, the analysts can formulate a new conjecture. When the player is done, the analyst updates the case to describe what she thinks has just transpired, and this is remembered so that, when a similar case arises in the future, it may be consulted. For visualization, we use the Google Earth API to render a KML file on a background of a satellite image and for scripting.

We use the FrameNet lexical database for frames that have been formulated by analyzing language corpora. The FrameNet project has recorded the information on frames in an OWL file. In IAGOA, a modeler, assisted by a program that takes as input

this file, develops form elements and associated script to query the analysts for verbs and the FEs of the associated frames. The analyst is presented with a display produced on a webpage using the Google Earth API. When the analyst clicks near the player, she will be presented with a list of verbs. After selecting a verb, she will be presented with the associated form, which will be filled in principally with values of a spatial and sometimes temporal FEs. The Google Earth API is used to identify spatial features.

The next chapter presents background, including techniques for representing objects and their relations qualitatively, linguistic notions and FrameNet, and the various software packages and frameworks used. Regarding the software, Jena is a Semantic Web framework used here for processing OWL documents. Google Earth, which renders geospatial information that is encoded in KML files, is used here for portraying activities on backgrounds consisting of satellite images of the Earth's surface. And XSLT is a stylesheet language for transforming XML documents. Chapter 3 presents the software that is under development. The next chapter discusses cognitive aspects of IAGOA, including how it facilitates identification of a player's intentions and enhances an analyst's situation awareness. Chapter 5 concludes.

CHAPTER 2

BACKGROUND

The first section in this chapter introduces linguistic notions that are important for this research as well as the FrameNet lexical database. It includes an analysis of two frames that are important in the current context to indicate the sort of analysis that underlies this work. The second section gives an overview of the Jena Semantic Web framework. It discusses the use of Jena with RDF documents, Jena's execution of SPARQL queries against RDF resources, and the use of Jena with OWL documents. The third section presents Google Earth, KML, and the Google Earth API. The last section introduces the XSLT stylesheet language for XML and presents an example relating to the lowercase semantic web.

2.1 Representing Objects and Qualitative Spatial Relations

The concern here is mainly with geospatial contexts and activities performed in such contexts. Metric notions here often require unreasonable precision and inhibit generalization, so qualitative (but no less mathematical) notions with more phenomenological plausibility are used. For representing objects in the real world, then, the perspective of Smith's mereotopology [4] is endorsed, which avoids point-set

topology's dependence on points that exist independently of all that is around them, and we accept Brentano's thesis that boundaries do not exist independently of the entities they bound. One thus introduces 1+Ds, which have significant length but whose width can generally be ignored, and 0+Ds, whose dimensions are so small that they are not considered in the model at hand. (In this nomenclature, a 0D is a mathematical point, a 1D a curve, and a 2D a region.) One, however, recognizes the importance of fiat objects, which are established by stipulation and are not under the same mereological constraints as natural objects. Fiat 0Ds (e.g., a stipulated starting point) as well as natural 1Ds (e.g., region boundaries) and fiat 1Ds (e.g., a line of advance) are also important and, in the case of the 1Ds, help demarcate 0Ds by intersection.

Turning to spatial relations, a semiorder is defined on 1Ds and 1+Ds, from which one can define an indifference relation. (Semiorders [5] are relations introduced in measurement theory to account for measurement where the order is not perfectly transitive.) One then defines indifference classes and a semiorder on these, which gives us an abstract qualitative notion of distance. To describe orientation and position qualitatively, Freksa's double cross calculus [6] is used, which partitions the plane into fifteen regions based on the relative positions based on the relative positions of a parent object (observer) and a reference object. This results in 15 ternary relations (ternary since they give the orientation of a third object relative to the parent and reference objects). By identifying given reference objects with new parent objects, one can define chains of these relations, which can be composed into relations of larger scope. We have shown [7] how the double cross calculus may provide upper and lower relative length bounds in

some circumstances. We have also shown how, using it, one can locate landmarks with respect to new orientations and can determine the locations of objects from new perspectives of players in a moving unit. One can also often make reasonably sharp conclusions about the relative locations of 1+Ds and even about the relative locations of regions.

2.2 Linguistic Notions and FrameNet

Given a word and universal grammar (UG) [8], the meaning of the word cannot be deduced without access to all essential knowledge that relates to that word. Theta theory (Θ-theory) mandates that verbs are referred to as predicates to include not only the verb but also its semantic categories, that is, thematic roles (Θ-roles). For example, a verb such as *hug* brings in two Θ-roles, an agent (subject participant) and patient (object participant).

To relate linguistic semantics to encyclopedic knowledge and utilize the concepts similar to those presented in Θ-theory, a large lexical database called FrameNet [9, 2] s used, which is accessible online at http://framenet.icsi.berkeley.edu/. FrameNet is based on Charles Fillmore's frame semantics [10], where a frame "is any system of concepts related in such a way that to understand any one concept it is necessary to understand the entire system …" [11]. These concepts (or categories) are called frame elements (FEs) in FrameNet, the equivalent of Θ-roles. FEs have inheritance relations, which eventually

lead to a semantic type; these types are related to SUMO (http://www.ontologyportal.org/) and themselves have inheritance relations. Although a given frame may be associated with several parts of speech (e.g., the verb *move* and the noun *motion* elicit the same frame), verbs are paramount in frame semantics, and restrict attention is restricted to them. The lexical database essentially consists of lexical units (LUs), which are pairings of words (verbs, nouns, adjectives, and adverbs) with frames. Note that one word may be paired with several frames (polysemy). For instance, the word "bake" is linked to three different frames. One is the *Apply_heat* frame, when used in such a way as "Michelle baked the potato for 45 minutes." Although the same verb is used in the sentence "Michelle baked her mother a cake for her birthday", the corresponding frame elements imply a different meaning of the verb, *baked*. In this case, the object is "mother" and therefore alters the semantic types worked with, and so this type requires a different frame (*Cooking_creation*). Lastly, there is the usage "The potatoes have to bake for more than thirty minutes", which evokes the *Absorb_heat* frame. The verb by itself does not have total meaning until it is placed into context with its frame elements. A word evokes a frame of semantic knowledge relating to the specific concept it highlights.

The verb data used for this research is obtained from FrameNet's OWL distribution [12], OWL being the Web Ontology Language [13]. An ontology is a formal, explicit specification of a shared conceptualization [14]. It provides a shared vocabulary, which can be used to model a domain — that is, the types of objects or concepts that exist and their properties and relations. Although ontology had its roots in philosophy, it is

applicable in computer science and, by extension, to the research presented here. What ontologies have in common in both computer science and in philosophy is the representation of entities, ideas, and events along with their properties and relations, according to a system of categories. These concepts can be utilized in computational areas by using standards such as OWL to represent knowledge.

In intelligence situations, two actions of great interest are *move* and *survey* (or *look for*). One may observe movement by players in an area of operation and build a sentence structure that provides for the predicate and the arguments that go along with the predicate. Further, one may then perform any necessary automated reasoning on the data once made observations are made. Consider the sentence *The terrorist is moving from here through there*. The Θ-roles are *Agent*, *Source*, *Goal*, *Direction*, and *Path*. The *Agent* is understood to be the target being tracked. The geospatial coordinates may be inserted in the sentence for the indexical terms *here* as the source location and *there* for the goal location. Given this sentence, the source coordinates serve as the first argument of the predicate (*Source*). The destination coordinates serve as the second argument of the predicate (*Goal*).

Although many verbs will be addressed, a closer look is taken at the two verbs mentioned above, *move* and *survey*, and the presentation in [15] is followed. Since one can be monitoring living beings or machinery, including manned and un-manned, that has a level of artificial intelligence or is remotely controlled by humans, the select verbs relate to activities of these types of entities.

The classification of verbs in Levin [16] is often used by the NLP community as

evidence of the semantic similarity of verbs. This classification is based on the alternations that phrases in which a given verb appears to undergo while preserving the meaning of the phrase. Many verbs are compatible with multiple argument frames in this sense, and those compatible with the same syntactic alternations are placed by Levin in the same class (and presumably share a common semantics). Levin's verb classifications have been seen as an alternative to FrameNet's frames [17], and taking both approaches into account at least gives the benefits of multiple perspectives.

The most popular lexical database, and the most influential for ontology development, is WordNet [18]. It groups words into sets of synonyms called synsets; a word belongs to as many synsets as it has meanings. Relations between synsets are justified psychologically by how humans process language and depend on the type of word. For example, nouns are related by is-a (subclass-of) and part-of relations and their inverses, and verbs are related by is-a, specialization by manner, and entailment. IAGOA makes use of FrameNet explicitly because a frame provides a constellation of roles that suggest what information to provide to fill out the description of an action.

2.2.1 Frame Semantics

In frame semantics [11], a frame "is any system of concepts related in such a way that to understand any one concept it is necessary to understand the entire system; introducing any one concept results in all of them becoming available." For example, The Commercial Transaction frame (whose FEs include a buyer, a seller, goods, and money) is related to a set of semantically related verbs, including *buy*, *sell*, *pay*, and *spend*, each evoking a different aspect of the frame. Knowing the meaning of any one

requires knowledge of commercial transactions hence in some sense knowing the meanings of the others. Frames are also characterized as experience-based schematizations of the speaker's world that impose order on prototypes, where a prototype is a segment of one's culture providing a backdrop for understanding a word. In contrast to semantic field theory, where a word is defined in relation to other words in the same "field" (i.e., those with which it contrasts), in frame semantics, a word is defined in relation to its background frame (and so there may be frames with single lexical representatives).

Much of the frame semantics literature addresses words, but frame semantics seeks a uniform representation for the meanings of words, sentences, and texts [19]. And it provides an account of various lexical, syntactic, and semantic phenomena. The interpreter of a text or discourse invokes a frame when assigning an interpretation to a piece of text by placing its contents in a pattern known independently of the text, and a text or discourse evokes a frame when a linguistic pattern is conventionally associated with that particular frame. Furthermore, a complete description of, for example, a verb must also include information about its grammatical properties and the various syntactic patterns in which it occurs. What FEs may be realized as subject or object and the surface forms of the others can be derived from a sufficiently rich description of the FEs.

Fillmore distinguishes U-semantics (the semantics of understanding) from T-semantics (truth-conditional semantics), that is, formal Tarskian semantics [19]. U-semantics determines what it takes to interpret text or discourse (the interpreter invoking or the text evoking a frame) whereas T-semantics addresses the truth conditions of a

sentence, where truth is determined compositionally. Fillmore holds that the data of 'understanding', not the "theory-defined derivative data limited to" truth conditions, is the phenomenologcially primary data for language theory. (This is not to say that T-semantics does not have its place.) Fillmore accepts that U-semantic is compositional in that building an interpretation relies on knowing the meanings of the individual words, phrases, and grammatical constructions, but it is non-compositional in that a sentence interpretation is not guided by purely symbolic operations from bottom to top.

Fillmore and his colleagues extended the representation for lexical items to represent grammatical constructions as well, giving the framework called construction grammar [11]. Argument structure constructions (such as [Subject Verb IndirectObject DirectObject], as found in, e.g., "Ed gives Sue the book") also invoke frames, so the meaning of a particular argument-structure construction derives from not just the meaning of the constituent verb but also the frame associated with the construction itself. The primary unit here is the grammatical construction, not syntactic atoms and composition rules, and the grammar of a language consists of taxonomies of families of constructions. According to most construction grammarians [20], a grammatical construction pairs form and content, and all such pairings are constructions, including sentences and words and structures of greater, less, and intermediate scope. The form covers not just syntax but also phonology, and the content covers not just semantics but also pragmatics. Gricean relevance and implicature involve not just pragmatics but also linguistic schematization [19].

Perspective is an important concept in frame semantics [11]. For example, "buy"

and “sell” evoke very similar frames with the same FEs but with different perspectives. The generalization here in FrameNet is the notion of frame-to-frame relations [2], one of which is the Perspective_on relation, so that the Commerce_buy and Commerce_sell frames have Perspective_on relationships with the neutral Commerce_goods-transfer frame. Frame-to-frame relations also include the familiar Inheritance relation and the Subframe relation, which holds when a complex frame refers to a sequence of states and transitions, each separately described as a frame; the Precedes relation holds among pairs of these subframes. Another relation is Causative_of, which holds when one frame indicates the cause of the state while the other only indicates the state (e.g., *John broke the jar* vs. *The jar broke*). An obvious enhancement to IAGOA would allow the analyst, after selecting a frame, to identify a sequence of associated subframes. Also, if circumstances warrant, we could try to lead the analyst from a frame to the related causative frame. In general, frame-to-frame relations provide the potential to dialog with the analyst to fill in aspects in the description of the situation at hand.

The notion of frame in frame semantics should be contrasted with that introduced by Minsky [21], which has become a standard AI representation. According to Minsky, a frame is a data-structure for representing a stereotyped situation. A frame has slots (or terminals), which may be filled with defaults that support expectations but in any case are filled by a matching process to represent the current situation. Frames are linked together into frame systems so that frames in the same system (e.g., representing different perspectives or cause-effect relations) share terminals. Frame systems in turn are linked by an information retrieval network that provides a replacement for a frame that fails to

match reality.

Minsky considered Fillmore's case grammar [10], which he developed before moving on to frame semantics. Case grammar sees a sentence as a combination of a verb plus a set of deep cases (i.e., semantic roles), such as Agent, Location, or Instrument. Each verb selects certain deep cases, some mandatory, some not. Grammatical functions (e.g., subject and object) are selected in a way that depends on the deep cases available, and deleting mandatory cases results in grammar violations. Minsky accepted verb-centered frame-like structures for analyzing sentences but held that such structures often become subordinate or even disappear in more extended discourse. As sentences are understood, we merge substructures into a growing frame if we can. Otherwise, a replacement frame is retrieved and previous assignments to terminals are reused if possible or, if not, a more radical replacement is sought. Minsky admitted that merging frames raises fundamental problems.

Neither case frames nor semantic frames, however, are intended to be hierarchical structures elaborated to mirror reality. Two frames can be combined by identifying the denotation of the filler of an FE in one frame with the denotation of a filler in the other, which is exactly what IAGOA does. In fact, this co-referential approach is one way to derive more complex frames from simpler ones. A given text or discourse segment may include a summary that evokes a frame of greater scope than those evoked by other parts of the segment, and instantiating the FEs of one frame may overlap the instantiation of the FEs of another frame, but neither case involves a hierarchy in the sense of Minky's frames. In addition, since a semantic frame is evoked by the text or discourse at hand, it is

rejected only when misunderstanding ensues (as with ambiguous constructions) and not because of a mismatch with the situation. Again, this is the policy of IAGOA: rejecting a conjecture is rejecting the phrase expressing it (and only thereby the associated frame).

2.2.2 Example Lexical Unit

2.2.2.1 Move Lexical Unit

Move is listed by Levin under multiple distinct verb classes. For example, slide verbs are a subclass of the parent verbs of sending and carrying. These are transitive and therefore not appropriate for an intelligence scenario, as we are considering verbs that can be associated with a phrase of the form *from location$_1$ to location$_2$*. The category of Levin's that we are indeed interested in is the category of Verbs of Inherently Directed Motion, which contains *move*. Levin's list of unquestionable members of this class is

> *advance , arrive, ascend, come, depart, descend, enter, escape, exit, fall, flee, go, leave, plunge, recede, return, rise, tumble*

Her comment on this class begins as follows. Note that to specify something in deictic terms means to rely on the context to convey meaning, as with indexical words like, *I*, *here*, and *now*.

> The meaning of these verbs includes a specification of the direction of motion, even in the absence of an overt directional complement. For some verbs this specification is in deictic terms; for others it is in nondeictic terms. None of these verbs specify the manner of motion. However, the members of this class do not behave uniformly in all respects. They differ as to how they can express the goal, source, or path of motion; depending on the verb, these may be expressed via a prepositional phrase, as a direct object, or both

Note the roles *direction*, *goal*, *source*, and *path*.

FrameNet defines the *Motion* frame as follows.

> Some entity (*Theme*) starts out in one place (*Source*) and ends up in some other place (*Goal*), having covered some space between the two (*Path*). Alternatively, the *Area* or *Direction* in which the *Theme* moves or the *Distance* of the movement may be mentioned.

Its theta-roles, referred to as frame elements in FrameNet-speak, are:

Theme: The entity that changes location. Note that it is not necessarily a self-mover.

Source: The location the *Theme* occupies initially before its change of location

Goal: The location the *Theme* ends up in

Path: (A part of) the ground over which the *Theme* travels or a landmark by which the *Theme* travels.

Area: The setting in which the *Theme*'s movement takes place without a specified *Path*.

Direction: This indicates motion along a line from the deictic center towards a reference point (which may be implicit) that is neither the *Goal* of the posture change nor a landmark along the way of the moving part of the body.

Distance: Any expression which characterizes the extent of the *Motion*.

Move evokes (triggers) the *Motion* frame and is represented in OWL as a class, namely *F_Motion*. This class has a number of restrictions and properties, but of particular interest are the frame elements represented in OWL as *FE_Direction_6370*, *FE_Distance_1965*, *FE_Goal_29*, *FE_Path_28*, and *FE_Source_27*.

2.2.2.2 Survey Lexical Unit

Survey belongs to the class of searching verbs. According to Levin, verbs pertaining to searching have three ways of expressing their arguments.

1. NP1 V NP2 *in* NP3
2. NP1 V NP2 *for* NP3
3. NP1 V *for* NP2 *in* NP3

(Here NP1, NP2, and NP3 are noun phrases, and V is a verb.) The verbs belonging to this class take as arguments the agent, the entity being sought, and the location where the search is performed. The entity that is being sought and the location where the search is performed may be expressed either as the direct object of the verb or within a prepositional phrase.

There are several children that belong to the class of searching verbs. For intelligence operations, the focus is on Investigate Verbs, which include verbs such as *explore, examine, inspect, scrutinize,* and *survey*. We select *survey* for our purpose.

In FrameNet, the verb *survey* evokes the *Scrutiny* frame, defined as follows.

> "This frame concerns a *Cognizer* (a person or other intelligent being) paying close attention to something, the *Ground*, in order to discover and note its salient characteristics. The *Cognizer* may be interested in a particular characteristic or entity, the *Phenomenon* that belongs to the *Ground* or is contained in the *Ground* (or to ensure that such a property of entity is not present). Some words in this frame allow alternate expressions of the *Ground* the *Phenomenon*. So, the frame elements of interest for the survey action are *Agent*, *Ground*, and *Phenomenon*, with sentence resembling *Survey* for NP" [2].

2.3 Jena

Jena [22] is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract model. A model can be sourced with data from files, databases, URLs or a combination of these. A model can also be queried through SPARQL and updated through SPARUL, an acronym for SPARQL/Update. (SPARUL is not currently a standard.) Jena also provides support for OWL. The framework has various internal reasoners and the third party Pellet reasoner (an open source Java OWL-DL reasoner) can be set up to work in Jena. Jena supports serialization of RDF graphs to a relational database, RDF/XML, Turtle, and Notation 3 (N3).

RDF [23] is the Resource Description Framework, and as the name implies, is a standard for describing resources. A resource is basically anything that can be identified and is represented by a Uniform Resource Identifier (URI). Consider the resource, `http://www.example.com/AlbertEsterline`. Graphically, a resource is represented as an ellipse (See Figure 2).

Figure 2. Graphical Representation of an RDF Resource

Each resource has a property, which is shown as an arc and representing a URI, as shown in Figure 3. The property contains a value, which may either be a URI or a literal. Assuming that the resource above is referring to Dr. Albert Esterline, the professor at North Carolina A&T State University, we can specify a triple with the full name of this resource being Albert Esterline. The subject (resource) in this example is the URI in the ellipse; the property is an arc containing a URI listed in XML qname form. Finally, the value of the property is a literal (“Albert Esterline”).

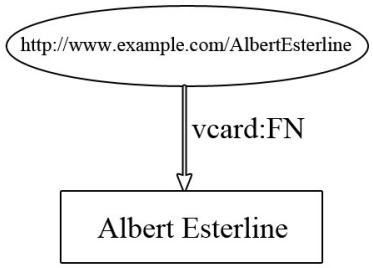

Figure 3. Graphical Representation of an RDF Triple

The Jena API has object classes to represent graphs, resources, properties and literals and can therefore represent the above and other RDF graphs as program code. The interfaces representing resources, properties and literals are called *Resource*, *Property* and *Literal* respectively. In Jena, a graph like the one above is called a model and is represented by the *Model* interface. Java code using the Jena API for this particular model

is shown in Appendix A, which includes a general discussion of Jena and RDF.

For RDF to be useful, we need some way of retrieving the data. One of the more popular methods is a query language called SPARQL. SPARQL is a query engine that comes packaged with the Jena API. The SPARQL syntax is similar to the SQL language commonly used in database queries but is pattern-based. See Appendix B for an overview of SPARQL.

An ontology model is an extension of the Jena RDF model that provides extra capabilities for handling ontologies. See Appendix C for a summary of processing OWL with Jena.

2.4 Google Earth and KML

The most popular and arguably most developed virtual globe is Google Earth, which is used in this project for portraying activities on backgrounds consisting of satellite images of the Earth's surface. Google Earth renders geospatial information that is encoded in KML files. The Google Earth API lets one make Google-Earth functionality available on custom-made web pages. This section gives a summary of these topics.

2.4.1 Google Earth

Google Earth [24] (see its homepage, http://earth.google.com/) is a virtual globe, map and geographic information program acquired by Google in 2004. It maps the Earth by the superimposition of images obtained from satellite imagery, aerial photography and

GIS 3D globe. Google Earth displays satellite images of varying resolution of the Earth's surface, allowing users to see features looking perpendicularly down or at an oblique angle with perspective. Most land is covered at a minimum of 15 meters of resolution. Google Earth allows users to search for addresses for some countries, enter coordinates, or use the mouse to browse to a location.

For large parts of the surface of the Earth, only 2D images from nearly vertical photography are available. For other parts, 3D images are available. Google Earth uses digital elevation model (DEM) data collected by NASA's Shuttle Radar Topography Mission (SRTM). Google Earth can also show 3D buildings and other structures portrayed in users' submissions done with SketchUp, a 3D modeling program. Google Street View provides a street-level view in many locations. For some locations, Google Earth can be used to monitor traffic speeds at loops located every 200 yards in real-time.

The internal coordinate system of Google Earth is geographic coordinates (latitude/longitude) on the World Geodetic System of 1984 (WGS84) datum. Google Earth shows the earth as it looks from an elevated platform.

Many use various applications to add their own data, making the results available through various sources, such as the bulletin board systems or blogs. Google Earth can show images overlaid on the surface of the earth and is a client for the Open Geospatial Consortium’s (OGC’s) Web Map Service (WMS) protocol for serving georeferenced map images over the Internet.

2.4.2 KML

Google Earth supports managing 3D geospatial data through the Keyhole Markup Language (KML) [25] (see its Google page, http://code.google.com/apis/kml/). KML is an XML-based language for expressing geographic annotation and visualization on Web-based, 2D maps and 3D Earth browsers. KML was created by Keyhole, Inc. (which was acquired by Google in 2004) and is now an OGC standard. Google Earth was the first program able to view and graphically edit KML files, and other projects such as Microsoft's Bing Maps have also developed KML support.

A KML file specifies a set of features (placemarks, images, polygons, 3D models, textual descriptions, etc.) for display in a 3D earth browser (geobrowser) implementing the KML encoding. Each place has a longitude and a latitude. Other data can make the view more specific, such as tilt, heading, and altitude, which together define a "camera view". One can use KML to specify icons and labels to identify locations on the planet surface. Different camera positions can be created to define unique views for each of several features. COLLADA textured 3D objects can be displayed. One can also use image overlays attached to the ground or screen, define styles to specify feature appearance, and write HTML descriptions of features, including hyperlinks and embedded images. Styles are set at the `Placemark` level, either using a `styleUrl` element or a `Style` element. Ground overlays let one "drape" an image onto the Earth's terrain; the `Icon` element contains the link to the `.jpg` file with the overlay image. The `Region` tag is used to control what features are being displayed to the user. The `MultiGeometry` element groups geometries together in the same `Placemark` element.

This allows the geometries to share the same styling and to appear as one item in a list such as the My Places pane in Google Earth. `Polygons` can be used to create simple buildings and other shapes.

The time slider in KML opens as soon as a KML file with a `TimeStamp` or `TimeSpan` element is opened. The time slider finds all currently selected `TimeStamp` and `TimeSpan` elements and adjusts to accommodate all dates represented. KML currently does not allow more control over the time slider, such as setting the current time selected, or selecting how wide a time span is represented. Currently, KML in Google Earth only supports a subset of HTML that describes presentation, not interaction.

KML shares some of the same structural grammar as the Geography Markup Language (GML) [26], the XML grammar defined by the OGC to express geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. Note that the OGC has also adopted KML and has published a standard for it [27].

One can create KML files with the Google Earth user interface, or one can use an XML or simple text editor. Placemarks, ground overlays, paths, and polygons can all be authored directly in Google Earth. There are also a few ways of creating KML automatically from a spreadsheet. Folders can be used for hierarchical grouping of features.

KML files are very often distributed in KMZ files, which are zipped files with a `.kmz` extension. The compression used must be legacy (ZIP 2.0) compatible (e.g., the deflate method), otherwise the file might not uncompressed in all geobrowsers. The

contents of a KMZ file are a single root KML document (notionally `doc.kml`) and optionally any overlays, images, icons, and COLLADA 3D models referenced in the KML file including network-linked KML files. The root KML document is typically a file named "`doc.kml`" at the root directory level. When a KMZ file is unzipped, a single *`doc`*`.kml` is found along with any overlay and icon images referenced in the KML.

Just as web browsers display HTML files, Earth browsers such as Google Earth display KML files. Once the server has been configured and the URL of the KML files has been shared, anyone with Google Earth installed can view the KML files hosted on a public web server. The MIME type associated with KML is `application/vnd.google-earth.kml+xml`, while that associated to KMZ is `application/vnd.google-earth.kmz`. Web servers have to be told what kinds of files they are providing. Some browsers, such as Firefox, will make guesses on the file type. Others, such as Internet Explorer, rely on the web server's MIME Type settings. For an Apache server, one would include these two lines in the `httpd.conf` configuration file:

```
AddType application/vnd.google-earth.kml+xml .kml
AddType application/vnd.google-earth.kmz .kmz
```

KML files can be dynamically fetched and updated from remote or local network locations, and KML data can be fetched based on changes in the 3D viewer.

Given the KML schema (http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd), a KML document can be validated with the usual XML editors, such as Oxygen. There are also online validators, such as FeedValidator.org (http://beta.feedvalidator.org/) and Galdos' KML Validator (http://www.kmlvalidator.com/), which show any errors and offer

style suggestions.

2.4.3 Google Earth API

It is often advantageous to make Google-Earth functionality available on custom-made web pages. The Google Earth Plug-in and its JavaScript API [28, 29] let one embed Google Earth into web pages. Using the API, one can draw markers and lines, drape images over the terrain, add 3D models, or load KML files.

To use the Google Earth API on a site, one needs a Google Maps API key; see [28] for how to obtain a key. Since the API is dynamically loaded via the Google AJAX API loader interface (`google.load`), the site will automatically use the latest revision for the requested major API version. After installing the plug-in, users will receive updates automatically and without interruption as they become available. Once the Earth plugin is working on the site, one can use the `google.earth.fetchKml` function to load any publicly accessible KML file.

The Google Earth API Reference [29] includes a description of the various interfaces, members, and `google.earth` functions in the Earth API. Interactive inheritance diagrams are included to depict relationships between interfaces. Interfaces whose names begin with `GE` allow for programmatic access to core plugin functionality and other miscellaneous options. Interfaces whose names begin with `Kml` represent KML-related objects such as `Placemark` and `LookAt`.

2.5 XSLT

XSLT [30] is a stylesheet language for XML. It is used to transform XML documents into other types of XML documents. For instance, a standard XML document marking up data may be transformed into an XHTML document for display. This is accomplished by matching patterns in the source XML document. Template rules may be applied recursively on the input, and an output based template can be specified for the desired destination. XSLT has many uses on the web, but this technology can be especially useful when working with the Semantic Web.

We consider an example using the lowercase semantic web. The lowercase semantic web extends existing XHTML for easily adding semantic data in human readable form. Web publishers can encode semantics into the HTML markup of web pages, allowing them to be consumed and processed. By adding some semantic markup to a web page that describes an upcoming concert, properties such as the date and location of the show can easily be extracted and used by other applications. A microformat is a web-based approach to semantic markup that seeks to re-use existing XHTML and HTML tags to convey metadata and other attributes.

GRDDL (Gleaning Resource Descriptions from Dialects of Languages) makes use of XSLT as a more efficient method of gathering data than a scraping solution. As the GRDDL acronym implies, there are number of dialects of XML (languages), and it is the function of GRDDL to translate and serialize the microformats of these XML dialects into RDF. This is achieved by providing a HTML metadata profile identifying the

document as a GRDDL source document and a link element to XSLT programs; external GRDDL-aware agents can then utilize this link for the extraction of data. By allowing the author of the source document the task of providing the XSLT, we avoid the conventional problem of screen scraping. With GRDDL, the publisher references a simple, reusable mechanism to extract relevant information [31]. An example here uses hCard markup.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head profile="http://www.w3.org/2003/g/data-view
          http://dublincore.org/documents/dcq-html/
          http://www.w3.org/2006/03/hcard
          http://gmpg.org/xfn/11">
  <link rel="transformation"
        href="http://www.w3.org/2006/vcard/hcard2rdf.xsl" />
  <title>Roland Johnson Bio</title>
</head>

<body>
  <div class="vcard">
    <a class="fn org url"
       href="http://www.rolandjohnson.org/">
      Roland Johnson personal website
    </a>
    <div class="adr">
      <span class="type">Work</span>:
      <div class="street-address">1019 Bent Branch St.</div>
      <span class="locality">Gastonia</span>,
      <abbr class="region" title="Gastonia">NC</abbr>
      <span class="postal-code">28054</span>
      <div class="country-name">USA</div>
    </div>
    <div class="tel">
     <span class="type">Work</span> 704-826-5494
    </div>
    <div>Email:
     <span class="email">rjohnson@ncat.edu</span>
    </div>
  </div>
</body>
</html>
```

Since OWL is an XML format, we may also leverage XSLT to transform OWL-based data into a form more useful. The `F_Motion` class shown below (from

FrameNet's OWL distribution) has a number of restrictions and properties, but of particular interest to us are specific frame elements. By gathering data with XSLT, upon extraction it may then be used at a convenient time and place for the user.

```
<owl:Restriction>
<owl:onProperty
    rdf:resource
    ="http://www.icsi.berkeley.edu/~jan/fnClassesTemplate.owl#hasFE"/>
<owl:someValuesFrom>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class
          rdf:about
       ="http://www.icsi.berkeley.edu/~jan/framenet.owl#FE_Direction_6470"/>
      <owl:Class
          rdf:about
       ="http://www.icsi.berkeley.edu/~jan/framenet.owl#FE_Distance_1965"/>
      <owl:Class
          rdf:about
       ="http://www.icsi.berkeley.edu/~jan/framenet.owl#FE_Goal_29"/>
      <owl:Class
          rdf:about
        ="http://www.icsi.berkeley.edu/~jan/framenet.owl#FE_Path_28"/>
      <owl:Class
          rdf:about
      ="http://www.icsi.berkeley.edu/~jan/framenet.owl#FE_Source_27"/>
    </owl:unionOf>
  </owl:Class>
</owl:someValuesFrom>
</owl:Restriction>
```

Shown below is the XSLT that gathers the frame elements in the above-mentioned `F_Motion` class. It discovers this motion class by matching based on the hierarchy of the FrameNet document. Once it locates the class and the associated frame elements, it loops through each one and in turn and transforms this data into an appropriate document for display, which is HTML here.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs
      ="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="xs"
        version="2.0"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
  <xd:doc xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl"
          scope="stylesheet">
    <xd:desc>
      <xd:p><xd:b>Created on:</xd:b> Jan 14, 2010</xd:p>
      <xd:p><xd:b>Author:</xd:b> Roland Johnson</xd:p>
      <xd:p></xd:p>
    </xd:desc>
  </xd:doc>
  <xsl:output method="html" />
  <xsl:template match="/">
    <xsl:apply-templates
       select
      ="/rdf:RDF/owl:Class[@rdf:about='http://www.icsi.berkeley.edu/~jan/framenet.owl#F_Motion']"
    />
  </xsl:template>
  <xsl:template
     match
      ="/rdf:RDF/owl:Class[@rdf:about='http://www.icsi.berkeley.edu/~jan/framenet.owl#F_Motion']">
    <html>
      <body>
        <xsl:if
           test
     ="/rdf:RDF/owl:Class[@rdf:about='http://www.icsi.berkeley.edu/~jan/framenet.owl#F_Motion']">
          <xsl:for-each
             select
        ="rdfs:subClassOf/owl:Restriction/owl:someValuesFrom/owl:Class/owl:unionOf" >
          <h3>
            <xsl:value-of select="owl:Class/@rdf:about" />
          </h3>
          </xsl:for-each>
        </xsl:if>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

CHAPTER 3

SOFTWARE FOR ANALYSTS AND MODELERS

IAGOA has a component for the analyst and a component for a modeler who is responsible for HTML and associated JavaScript that allow the analyst to capture her current conjecture of the player's behavior in terms verbs and the FEs associated with the frames evoked by these verbs. The most significant input to the modeler software is the FrameNet OWL file, and the interface for the analyst and, to some extent, the modeler is provided by the Google Earth API.

The analyst works on a client. The server provides HTML documents that fetch updated KML documents. The server updates a KML document according to form data sent from the client. It also constructs HTML documents that provide form data according to the verbs and FEs currently selected for the case at hand. The analyst can track the player and, at the same time, watch her conjecture of what the player is up to. If the player's behavior deviates significantly from what has been conjectured, she can modify her conjecture or make a completely new conjecture. When a case being followed reaches its conclusion, the analyst updates her conjecture to her understanding of what has taken place and saves the case so that it may be accessed to provide insight when a similar case arises in the future.

The content of the initial KML documents is described first. The design for the analyst's software is then presented before that for the modeler's software since the

modeler's software makes sense only in that context.

3.1 Initial KML Files

IAGOA begins with a set of KML documents representing possible areas of operation that have been prepared in a way that facilitates the sort of analysis intended. The features (0+Ds, 1+Ds and 2Ds) of interest are encoded as KML `Point` elements (0+Ds), `LineString` elements (1+Ds), and `Polygon` elements (2Ds). Fiat 0Ds and 1Ds (such as a starting point or a property boundary) are handled similarly. (Note that KML allows only straight line segments, and the only 2D geometries are polygons; curvilinear features must be approximated by line segments.) Typed name-value pairs are added to the content of these elements using KML `ExtendData` elements. These appear in the balloon when the corresponding icon is clicked and include such attributes as the height and other salient properties of a building (a 1+D), the width and surface type of a road (a 1+D), and the dominant climax tree species of a forest (a 2D). KML styles are used so that the designated features are readily apparent, and different styles can be used to reflect different values for key attributes.

3.2 The Analyst

The analyst is presented with a display produced on a webpage using the Google Earth API. It is assumed that data will be provided (say, from a satellite) that updates in near real-time the location of a player of interest[1]. When the analyst clicks near the player, she will be presented with a list of verbs. Each verb represents an activity that she might conjecture currently applies to the player. She selects one verb at a time; additional verbs can be selected after information for the selected verb is submitted and the server returns a new HTML document that fetches an updated KML document (as mentioned below).

After selecting a verb from the list, she will be presented with the associated form, which will be filled in, principally with values of a spatial and sometimes temporal nature. The Google Earth API will be used to identify spatial features. With a given form element in focus, the analyst will click the feature on the KML rendering to get its id. If she wishes to select a 1+D that is not already defined in the KML document, she can define it by clicking a sequence of 0+Ds. Similarly, she can define a missing 2D with 1+Ds (or 1Ds) and, if necessary, 0+Ds (or 0Ds). If need be, she can define landmarks (0+Ds) on the fly, which will be sent along with the form data when the form is submitted. Some FEs come in alternative sets. For example, *Motion* is described in

[1] Google Earth allows track data to be imported from some GPS devices—see http://earth.google.com/support/bin/static.py?hl=en&page=guide.cs&guide=22373&topic=22374. Also, some companies provide spatial ETL (extract, transform and load) technology for interoperability of spatial data. See, e.g., Safe Software's Spatial ETL: http://www.safe.com/technology/spatialETL/overview.php.

terms of not only the *Source* but also either the *Path* and *Goal* or the *Direction* and *Distance*. Time is ubiquitous. Since we are talking about activities, the relevant notion is that of time period or (better, since the period is occupied by an activity) *Duration*. In the case of *Motion*, if *Distance* is somehow indicated, then one can deduce the *Duration* from the *Speed*.

Some properties that are neither spatial nor temporal cannot be ignored, such as the *Phenomenon* of interest with "survey". For such aspects, the analyst is presented with a menu of ontological categories. From the point of view of the player, the phenomenon cannot be identified with a location (otherwise the search would be pointless). The analyst, however, might be able to identify a location. In general, a non-spatiotemporal FE can optionally be assigned a location. If there is no reasonable location, it can generally be associated with the *Theme* (or *Cognizer* or *Actor*) and sometimes with some spatially global FE (such as the *Ground* or *Area*).

When the form is submitted, the server uses Jena to update a copy of the KML file. It identifies the instantiations of the FEs by associating with them `ExtendData` elements that identify what FEs they instantiate. Any new landmarks are added to the KML file. The file returned as the response to the request is an HTML document that fetches the new KML file. The player is again tracked on the rendering of this document, but now the conjectured change over time (such as change of position for *Move* or change in direction for *Survey*) is also shown. Also, the analyst may select another verb from the menu. In that case, it would generally be desired that at least one FE (other than that filled by the player) associated with the new verb be identified with an FE (of the same dimension or

ontological type) associated with a previously selected verb. This in effect defines a complex activity from two basic activities. For example, one could define an activity of moving toward a target that is being shot at by identifying the *Goal* FE of the *Motion* and *Use_firearm* frames.

A special case is where one or more FEs are instantiated with frames (as illustrated with Figure 2). The analyst selects the value for such an FE from the menu of verbs, thereby selecting the associated frame. The server's responses then present in succession the forms for the FE-instantiating frames without the analyst selecting the corresponding verbs.

The new HTML document has buttons used in case the analyst decides to modify her conjecture. There are buttons for each verb and each FE associated with each verb's frame, allowing the analyst to remove an FE or entire frame, replace it, or modify it. There is also a button that returns the original KML document, throwing away all the FrameNet-related information that has been added.

When a case that the analyst has been following is resolved, she saves this case so that it may be consulted for suggestions when a similar case arises in the future. The current design calls for not saving conjectures that turned out to be false, although future enhancements could save falsified conjectures since they may help an analyst avoid a garden path. The program allows the analyst to record some of her understanding of the case just concluded. When the analyst saves the final state, she selects (using the Google Earth API) those additional features deemed significant. They are marked with `ExtendData` elements, which can include annotations that explain why they are

important. The analyst can also indicate which differences in length are significant and which are not. To do so, she repeatedly selects two pairs of 0Ds or 0+Ds, representing two line segments, s_1 and s_2, whose lengths are to be compared: either s_1 is judged longer than s_2, s_2 is judged longer than s_1, or she is indifferent. Segments connecting 0+Ds or 0Ds that are not explicitly compared are ordered as determined by the values in the KML file. The ordering is stored in an `ExtendData` element in the KML file.

For the final state, it is often no longer acceptable to associate the speed or duration of the activity with the player. The progress of certain activities can be shown by moving an indicator on the KML rendering of the area of operation. For example, motion can be shown as the player changing positions over time, and looking at (frame *Perception_active*) can be shown as the direction changing over time. This can be captured in KML using `TimeStamp` elements with the elements representing the entity at different locations at different times. Generally, only locations corresponding to marked landmarks will be used, although it might be necessary to define new landmarks (such as bends in a road). This time-varying information can be captured in several ways, such as by having the analyst direct the program with clicks as the case evolves, by automatically saving values of GPS data when they coincide with landmarks, or by having the analyst fill in details after the fact.

3.3 The Modeler

The modeler, using FrameNet resources and with the help of several programs, constructs, for each verb the analyst may choose, the HTML document whose form is presented by the server to the analyst. The same HTML document is used with any number of KML documents representing areas of operation.

XSLT is used to extract the lexical units and associated frames from the FrameNet HTML documents that contain these associations. This produces an HTML document. On rendering this HTML document, the modeler will select the lexical unit he wishes to capture. If there is more than one frame associated with that lexical unit, then he must select the particular frame; the analyst can view the FrameNet HTML pages to see the definitions of the various frames. Selecting a frame will invoke a Jena program that processes the FrameNet OWL file, starting with that frame. The modeler will be presented with one FE after another associated with that frame. He will select the FEs of interest (again referring to the HTML pages for explanations). For each selected FE, the Jena program will chase back the inheritance relations until it gets to the semantic types (e.g., *EM_Locative_relation_182*), and the modeler will have the option to following the inheritance relations among semantic types to get to more fundamental semantic types.

For a given frame, what FEs are provided and which are required and in what combination are questions that must be addressed and can result in considerable structure in the choices presented to the analyst. According to [2, sec. 3.2.1], a core FE normally is instantiated with its frame, but there are numerous cases where it may be omitted. As

noted, some FEs form alternative sets. Also, a core FE may be "null instantiated" [2, sec. 3.2.3], as when the element is understood in the context; note that the FE instantiated with the player can generally be null instantiated in this sense. A core FE is also null instantiated when the semantic type of the missing element is clear from conventions (as when a transitive verb is used intransitively, as in *Jim eats alone*) or the omission of a constituent is licensed by a grammatical construction (e.g., omitting the agent in a passive sentence). There are also several relations among FEs [2, sec. 3.2.2]. Sometimes the occurrence of one core FE requires the occurrence of another, while some FEs are incompatible. Finally, if a group of FEs forms a "coreness set," then anyone of these FEs may stand alone; this is the case, for example, with the FEs *Source*, *Path*, and *Goal* of the *Motion* frame.

The Jena program is being used to construct the HTML form from the selected FEs with considerable help from the analyst. The modeler will control the explanatory text associated with each form element. If the semantic type of an FE is spatial, then the form will be equipped with the code required to capture mouse clicks on the Google Earth display. If the semantic type of an FE is a duration or speed, then the form will be equipped with appropriate text boxes and menus for the units, and appropriate validation code will be included. If the semantic type of an FE is non-spatiotemporal, then the modeler will populate a menu with values consistent with the semantic type. (Lists of values for various semantic types could be drawn up in advance.) The modeler will also have the option of including a parameter for the id of a KML geometric element associated with a non-spatiotemporal FE.

The Jena program must also produce HTML fragments and corresponding JavaScript code that the server uses in constructing HTML documents that form the response when the analyst updates the current case. These are simply fragments of the HTML document discussed in the previous paragraph.

CHAPTER 4

COGNITIVE IMPLICATIONS

Since frame semantics characterizes word meaning in terms of experience-based schematizations of the speaker's world, IAGOA has a basis in human cognition, and it is designed to work with an analyst in a way commensurate with human cognition. With its linguistic foundation, IAGOA addresses literally the comprehension of the meanings of elements in the environment, and it addresses the projection of the situation in the near future. These are hallmarks of situation awareness, and, indeed, IAGOA's main goal can be expressed in terms of situation awareness. Given the salient geospatial features the player experiences and their relations, IOGOA allows one to address the possibilities open to the player and to index relevant previous cases, topics also covered in the first section below. The next (and last) section in this chapter summarizes how this research fits squarely in the situation-awareness research program.

4.1 Intentions and Possibilities Open to the Player

This scheme will be enhanced to allow more revealing monitoring of movements of players. Now, note that the description of a player's activity provided by a verb frame generally indicates an intention. According to BDI principles [32], intentions are stable

(yet, for rationality, not irrevocable), supporting deliberation in advance of conduct. An intention, however, is generally partial; means-ends reasoning fills in the details, that is, forms a plan, which, for spatial behavior, is generally a path plan. The interest, however, is more in the intention, which is more abstract than the plan and in this respect more aligned with qualitative representations. To anticipate how a player might behave in a case, one moves out the activity horizon to allow significant features to fall within it and form a graph (with user assistance) from binary relations between the 1Ds and 1+Ds. Often an edge indicates that the player can move directly from one node to an adjacent node. An edge, however, could denote other things, for example, that the landmark denoted by an adjacent node is observable from the location associated with the given node. From a priori knowledge of the behaviors of players with the conjectured intention(s), one can conjecture what the area of operation is from the point of view of the player. One transforms the graph to express this point of view, giving what we call the intention graph. This often would be a subgraph of the original, but there may be reason to add nodes or edges to the original. The intention graph is related to the conjectured verb frames(s) in that FEs are instantiated with nodes or subgraphs of the intention graph. One can analyze the intention graph (e.g., regarding connectivity), possibly reduce it, and identify key parts (e.g., bridges in the graph-theoretic sense) to understand the possibilities open to the player. Given the player's intention and weak rationality assumptions, one anticipates the general form of the player's movement and its in-course changes.

After the user has described the current case, the system will attempt to find previous

cases that reasonably match it. The principal interest is in matching cases where each case describes a set of activities (using FrameNet-derived verb frames) glued together by pairs of NPs denoting the same geospatial features. Aspects of the case significant for matching are the classification of the player, the verb, the spatial relations among the denotations of the noun phrases (NPs) in the frame, and the classification of these denotations. The matching is expedited by using qualitative rather than metric spatial information (as outlined in [7]). A case would also include a skeletal representation of a geospatial region where the activities play out. Some kind of XML structure is assumed that is generally valid with respect to some schema; geospatial data would probably be in a form close to KML. Another scenario for matching is when, given an appropriately modified FrameNet verb frame and various constraints, we try to match the NPs in the frame with features in a geospatial region.

Several kinds of matching are relevant here. There is a significant literature on matching geospatial schemas (e.g., see the several papers on this in [33]). Tree and graph searching (cf., e.g., [34]) relates to finding one structure in another, and the paradigmatic case is trying to find a query (generally interpreted as a tree) as embedded in an XML document (as per the DOM). In schema-based matching (or just schema matching, cf. the survey in [35]), the schemas can be XML Schemas, database schemas, or ontologies (often expressed in OWL). The matching may be instance-based (as when a linguistic corpus is analyzed) but is usually based on the schemas themselves, where it is often used when joining two schemas. Schema matching can return several kinds of results, including a similarity measure but more often an alignment (of parts in one schema with

parts in another).

4.2 Situation Awareness

IAGOA directly addresses the situation awareness (SA) of the analyst and explicates some of the key notions of SA in well-defined, primarily linguistic terms. According to Endsley [36], SA is 'the perception of the elements in the environment …, the comprehension of their meaning and the projection of their status in the near future.' IAGOA takes "meaning" here literally as it uses the semantics of verbs describing the current situation, and projection in the near future is exactly the conjecturing that IAGOA requires. Endsley recognizes three levels of SA, the first being perception of cues, and the third being projection. The second level of SA is comprehension, "the integration of multiple pieces of information and a determination of their relevance to the person's goals." In the context of IAGOA, understanding an actor's goals follows from understanding his current activity, captured by one or more verb frames. Integration is provided by frames in a non-compositional process of instantiating frame elements, which exposes the relevance of the information to the player's goals since the verb frame schematizes his current activity.

While verb frames are not sufficient to capture all aspects of SA awareness, especially those of broad scope and those of a principally perceptual nature, they provide a clear explication for such central notions as schemas (in the form of constructions),

mental models, pattern matching, expectations, and goal-driven processing. Verb frames give a handle on natural language, which pervasively structures human experience and is particularly appropriate when the focus of the situation is another agent, where frames have not only descriptive but also generative force.

CHAPTER 5.

CONCLUSION

The work developed here, IAGOA (Intelligence Analyst's Geospatial and Ontological Assistant), helps an analyst classify the behavior of a player in an area of operation, to formulate clearly a case, and to find similar cases handled by other analysts. Qualitative spatial representations are used as well as the ontology of activities based on verb frames as proposed here. Analysis may be accomplished by retrieving simulated satellite data of ground vehicles and interacting with software modules that allow the analyst to conjecture the activities in which the actor is engaged along with the features of the area of operation relevant to the natures of those activities. Activities are conceptualized by ontologies. The research relies on natural language components (semantic frames) gathered from the FrameNet lexical database, which captures the semantics of lexical items with an ontology using OWL. The software has two components, one for the analyst, and one for a modeler, who produces HTML and parameterized KML documents used by the analyst. The modeler is responsible for code that allows the analyst to capture her current conjecture of the player's behavior in terms verbs and the FEs (frame elements) associated with the FrameNet frames evoked by these verbs. The most significant input to the modeler software is the FrameNet OWL file, and the interface for the analyst and, to some extent, the modeler is provided by the Google Earth API, used to render KML files on a background of a satellite image of the targeted

part of the Earth's surface and for scripting. The analyst identifies the fillers for the FEs and the system simulates the player and also updates the display showing what really transpires. If the conjecture deviates sufficiently from reality, the analysts can formulate a new conjecture (possibly involving a new verb). When the player is done, the analyst updates the case to describe what she thinks has just transpired in terms of the verb frames and what instantiated their FEs. This case is remembered so that, when a similar case arises in the future, it may be consulted for suggestions. Analysts will collectively develop taxonomies of tags in classifying objects in the area of operation, and graph theoretical analysis of the neighborhood of a player will help in understanding how it might meet its intention. Cases similar to the current will be found by an enhanced version of schema matching, and graph theory will be used to analyze relations within the area of operation. This effort addresses the higher levels of data fusion with formalisms that are intuitively meaningful and enhances the analyst's situation awareness.

The broader significance of this work is in the development of ontologies in general, which (one conjectures) ultimately rest on a dual basis of mathematical concepts and concepts extracted from natural language. Natural language must be taken into account in developing ontologies since most of the activities we wish to address have a social aspect that rests on the language used not only to describe but also to carry out the activities. The activity-related concepts necessarily involve thematic roles, or frame elements, as in frame semantics. There arises the question of how to determine the frames that are evoked in the use of a natural language. The most satisfactory answer to this is by analysis of language in action. A generally satisfactory approximation to such

an analysis is the analysis of language corpora, as is done with FrameNet.

The aspects of IAGOA not covered in this paper mostly relate to the use of qualitative representations of spatial properties and relations. (See the presentation in [7].) By avoiding exact numerical values, qualitative representations support more efficient spatial reasoning and judgments of similarity between cases than do quantitative representations. IAGOA uses such techniques for finding previous cases similar to the current case. It will also use qualitative techniques to gain insight into the player's perspective and opportunities and hence to assist the analyst in divining the player's intention and capturing her understanding of the case.

FrameNet provides suggestions for numerous enhancements to IAGOA. For example, it uses several frame-to-frame relations to situate frames in semantic space [2, chap. 6]. These relations include the familiar Inheritance relation (although FEs do not always retain the same names across this relation). They also include the Subframe relation, which holds when a complex frame refers to a sequence of states and transitions, each separately described as a frame; the Precedes relation holds among pairs of these subframes. An obvious enhancement to IAGOA would allow the analyst, after selecting a frame, to identify a sequence of associated subframes. Another relation is Causative_of, which holds when one frame indicates the cause of the state while the other only indicates the state (e.g., *John broke the jar* vs. *The jar broke*). In IAGOA, if circumstances warrant, we could try to lead the analyst from a frame to the related causative frame.

Among the aspects of a case that are beyond the scope of a system like IAGOA

without major enhancement, the most notable is the handling of multiple players. If the players are all part of a unit, then techniques for representing hierarchical organizations could be exploited [37]. If, in contrast, the players are opponents, then more analytically intensive techniques (such as game theory) are required.

With its linguistic foundation, IAGOA addresses the comprehension of the meanings of elements in the environment, and, in representing ongoing activities, it addresses the projection of the situation in the near future. Integration of multiple pieces of information according to frame semantics is a non-compositional process of instantiating frame elements, which exposes the relevance of the information to the player's goals since the verb frame schematizes his current activity. Finally, it is suggested that many of the central concepts of situation-awareness research can be fruitfully understood in frame semantics as utilized in IAGOA.

BIBLIOGRAPHY

[1] Baker, C. and Ruppenhofer J., "FrameNet's Frames vs. Levin's Verb Classes." In Larson, J., and Paster, M. (Eds.), *Proc. 28th Annual Meeting of the Berkeley Linguistics Society*, 2002, pp. 27-38.

[2] Basu, C., Cheng, H., and Fellbaum, C., "Creating a Geospatial and Visual Information Ontology for Analysts." In Hornsby, K. S. (Ed.), *Ontology for the Intelligence Community: Towards Effective Exploitation and Integration of Intelligence Resources*, Proc. 2nd Int. Ontology for the Intelligence Community (*OIC-2007*), Columbia, MD, 2007

[3] Boas, H., "From Theory to Practice: Frame Semantics and the Design of FrameNet." In Langer, S. and Schnorbusch, D. (Eds.), *Semantisches Wissen im Lexikon*, Tübingen, Germany: Gunter Narr Verlag, 2005, pp. 129-160.

[4] BouSaba, C., Esterline, A., Homaifar, A., and Fatehi, F., "Spatial Ontologies for Tactical Behaviors," *Proc. of the Unmanned Systems Technology X Conf.*, Orlando, FL, March 2008.

[5] Bratman, M. E., *Intention, Plans, and Practical Reason*. Stanford, CA: CSLI Publications, 1999.

[6] Cox, S., Daisey, P., Lake, R., Portele, C., and Whiteside, A., *OpenGIS® Geography Markup Language (GML) Implementation Specification*, version: 3.1.1, Open Geospatial Consortium, 2007, http://portal.opengeospatial.org/files/?artifact_id=4700.

[7] Davis, C. A., and Monteiro, A. M. V. (Eds.), *Advances in Geoinformatics*, New York: Springer-Verlag, 2007.

[8] Defense Geospatial Information Working Group (DGIWG), *DGIWG Feature and Attribute Data Registry*, https://www.dgiwg.org/FAD/, accessed May 2010.

[9] Endsley, M. R., "Theoretical Underpinnings of Situation Awareness: A Critical Review." In Endsley, M.R., and Garland, D.J. (Eds.), *Situation Awareness Analysis and Measurement*, Mahwah, NJ: Lawrence Erlbaum Associates, 2000, pp. 3-18.

[10] Esterline, A. and BouSaba, C., "An Ontology for Tactical Behaviors Derived from Verb Frames," *Proc. of the Unmanned Systems Technology X Conf.*, Orlando, FL, March 2008.

[11] Esterline, A., Johnson, R., Carr, E., and Wright, W., "An Analyst's Geospatial and Ontological Assistant," in *Proc. 2nd Int. Conf. on Software, Service & Semantic Technologies*, Varna, Bulgaria, 2010, pp. 89-96.

[12] Fellbaum, C. (Ed.), *WordNet: An Electronic Lexical Database*, Cambridge, MA: The MIT Press, 1998.

[13] Fillmore, C. J., "The Case for Case." In Bach, E. and Harms, R. T. (Ed.), *Universals in Linguistic Theory.* New York: Holt, Rinehart, and Winston, 1968, pp. 1-88.

[14] Fillmore, C. J., "Frames and the Semantics of Understanding," *Quaderni di*

Semantica, vol. 6, no. 2 (Dec. 1985), pp. 222-254.

[15] Freksa, C., "Using Orientation Information for Qualitative Spatial reasoning," *Proc. of the Int. Conf. on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Pisa, Italy, 1992, pp. 162-178,.

[16] GeoRSS Community, Main Page – GeoRSS, 2009, http://georss.org/

[17] Gibson, J. J., *The Ecological Approach To Visual Perception*, Hove, UK: Psychology Press, 1986

[18] Goldberg, A., *Constructions at Work: The Nature of Generalization in Language*. Oxford, UK: Oxford U. Press, 2006.

[19] Google, *Google Earth API Developer's Guide*, http://code.google.com/apis/earth/documentation/, accessed May 2010.

[20] Google, *Google Earth API Reference*, http://code.google.com/apis/earth/documentation/reference/, accessed May 2010.

[21] Google Earth Team, *Google Earth User Guide* (for Google Earth Version 5 and later), Google, 2010, http://earth.google.com/support/bin/static.py?page=guide_toc.cs.

[22] Grenon, P. and Smith, B., "SNAP and SPAN: Towards Dynamic Spatial Ontology," *Spatial Cognition & Computation*, 4:1 (2004), pp. 69-104.

[23] Guarino, N., "Formal Ontology and Information Systems." In Guarino, N. (Ed.), *Formal Ontology and Information Systems*, Amsterdam: IOS Press, 1998.

[24] ISO/TC 211 homepage, ISO, http://www.isotc211.org/, accessed May 2010

[25] Jena Development Team, "Jena Semantic Web Framework," http://openjena.org/, accessed May 2010.

[26] Kay, M. (Ed.), *XSL Transformations (XSLT) Version 2.0*, World Wide Web Consortium (W3C), http://www.w3.org/TR/xslt20/.

[27] Laboratory for Applied Ontology, "DOLCE," Trento, Italy: Institute of Cognitive Science and Technology, 2010, http://www.loa-cnr.it/DOLCE.html.

[28] Levin, B., *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago: U. of Chicago Press, 1993.

[29] Lymba Corporation, AFRL-RI-RS-TR-2009-122, Final Technical Report; *Synergist: Collaborative Analyst Assistant*, Rome, New York: Air Force Research Laboratory, Information Directorate, Rome Research Site, April 2009, http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA499605.

[30] Minsky, M., "A Framework for Representing Knowledge," in Winston, P. (Ed.), *The Psychology of Computer Vision*, New York: McGraw-Hill, 1975.

[31] Niles, I., and Pease, A., "Towards a Standard Upper Ontology." In Welty, C. and Smith, B. (Eds.), *Proc. 2nd International Conference on Formal Ontology in Information Systems* (*FOIS-2001*), Ogunquit, Maine, October 17-19, 2001.

[32] OGC (Open Geospatial Consortium), *Geography Markup Language*, 2010, http://www.opengeospatial.org/standards/gml.

[33] OGC (Open Geospatial Consortium), *KML*, 2010,

http://www.opengeospatial.org/standards/kml.

[34] Ouhalla, J., *Introducing Transformational Grammar: From Principles and Parameters to Minimalism*, 2nd ed., London: Hodder Arnold, 1999.

[35] Petruck, M., "Frame Semantics." In Verschueren, J., Östman, J-O., Blommaert, J., and Bulcaen, C. (Eds.), *Handbook of Pragmatics*, Philadelphia: John Benjamins, 1996.

[36] Powers, S., *Practical RDF*, Sebastopol, CA: O'Reilly, 2003.

[37] RDF Working Group, *RDF - Semantic Web Standards*, World Wide Web Consortium (W3C), 2010, http://www.w3.org/RDF/.

[38] Ressler, J. and Dean, M., "Geospatial Ontology Trade Study." In Hornsby, K. S. (Ed.), *Ontology for the Intelligence Community: Towards Effective Exploitation and Integration of Intelligence Resources*, Proc. 2nd Int. Ontology for the Intelligence Community (*OIC-2007*), Columbia, MD, 2007.

[39] Roberts, F. S. and Suppes, P., "Some Problems in the Geometry of Visual Perception," *Synthese*, vol. 17 (1967), pp. 173-201.

[40] Ruppenhofer, J., Ellsworth, M., Petruck, M. R. L., Johnson, C. R., and Scheffczyk, J., *FrameNet II: Extended Theory and Practice*, 2006, Web Publication, http://framenet.icsi.berkeley.edu/book/book.html.

[41] Scheffczyk, J., Baker, C. F., and Narayanan, S., "Ontology-based Reasoning about Lexical Resources," *Proc. Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies* (*OntoLex 2006*), Genoa, Italy, 2006, pp. 1-8.

[42] Self, T., Kolas, D., and Dean, M., "Ontology-Driven Imagery Analysis." In Hornsby, K. S. (Ed.), *Ontology for the Intelligence Community: Towards Effective Exploitation and Integration of Intelligence Resources*, Proc. 2nd Int. Ontology for the Intelligence Community (*OIC-2007*), Columbia, MD, 2007.

[43] Shasha, D., Wang, J. T. L., and Giugno, R., "Algorithmics and Applications of Tree and Graph Searching," *Proc. Symp. on Principles of Database Systems (PODS)*, Madison, WI, 2002, pp. 39–52.

[44] Shvaiko, P. and Euzenat, J., "A Survey of Schema-Based Matching Approaches," *Journal on Data Semantics*, vol. 4 (2005), pp. 146-171.

[45] Sletten, B., "Gleaning Information from Embedded Metadata," *DevX.com*, February 7, 2008, http://www.devx.com/semantic/Article/36973/0/.

[46] Smith, B., "Mereotopology: A Theory of Parts and Boundaries," *Data and Knowledge Engineering*, vol. 20 (1996), pp. 287-303.

[47] Smith, B., Vizenor, L., and Schoening, J.," Universal Core Semantic Layer." In Costa, P., Laskey, K., and Obrst, L. (Eds.), *Proc. 2009 Int. Ontology for the Intelligence Community* (*OIC-2009*), Fairfax, VA, 2009.

[48] Smith, M., Welty, C., and McGuinness, D., *OWL Web Ontology Language Guide*, W3C Recommendation, 2004,. http://www.w3.org/TR/owl-guide/.

[49] *UCore 2.0: Breaking Barriers to Information S*haring, https://ucore.gov/ucore/, retrieved May 2010.

[50] Wernecke, J., *The KML Handbook: Geographic Visualization for the* Web, Reading, MA: Addison-Wesley, 2008.

APPENDIX A

USING JENA WITH RDF

Recall that, in Jena, a graph like the one shown in Figure 3 is called a model and is represented by the *Model* interface. Jena code using the Jena API for this particular model follows. The code is rather straightforward. It begins with some constant definitions and then creates an empty model object using the `ModelFactory` method `createDefaultModel()` to create a memory-based model. The `albertEsterline` resource is then created and a property added to it. The property is a "constant" class `VCARD`, which holds objects representing all the definitions in the `VCARD` schema. Jena provides constant classes for other well known schemas, such as RDF and RDF schema themselves, Dublin Core and DAML.

```
// some definitions
static String personURI    = "http://www.example.com/AlbertEsterline";
static String fullName     = "Albert Esterline";

// create an empty Model
Model model = ModelFactory.createDefaultModel();

// create the resource
Resource albertEsterline = model.createResource(personURI);

// add the property
albertEsterline.addProperty(VCARD.FN, fullName);
```

We may also write equivalent source code using the cascading style:

```
// some definitions
static String personURI    = "http://www.example.com/AlbertEsterline";
static String fullName     = "Albert Esterline";

// create an empty Model
```

```
Model model = ModelFactory.createDefaultModel();

// create the resource and add the property
Resource albertEsterline =
        model.createResource(personURI)
             .addProperty(VCARD.FN, fullName);
```

Jena has an extensible interface which allows new writers for different serialization languages for RDF to be easily plugged in.

Complete working source of the above code, writing the output in N3

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.vocabulary.*;

/* Example1 creating a simple model */
public class Example1 extends Object {

    // some definitions
static String personURI    = "http://www.example.com/AlbertEsterline";
static String fullName     = "Albert Esterline";

      public static void main (String args[]) {

// create an empty Model
Model model = ModelFactory.createDefaultModel();

// create the resource and add the property
Resource albertEsterline =
        model.createResource(personURI)
             .addProperty(VCARD.FN, fullName);
      }

model.write(System.out, "N-TRIPLE");
}
```

Complete working source of the above code, writing the output in RDF/XML

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.vocabulary.*;

/* Example1 creating a simple model */
public class Example1 extends Object {

    // some definitions
static String personURI    = "http://www.example.com/AlbertEsterline";
static String fullName     = "Albert Esterline";

      public static void main (String args[]) {

// create an empty Model
Model model = ModelFactory.createDefaultModel();
```

```
// create the resource and add the property
Resource albertEsterline =
        model.createResource(personURI)
             .addProperty(VCARD.FN, fullName);
      }

model.write(System.out, "RDF/XML-ABBREV");
}
```

The Jena model interface defines a `listStatements()` method, which returns a `StmtIterator`, which has a method `nextStatement()`. This method returns the next statement from the iterator. The `Statement` interface provides accessor methods to the subject, predicate and object of a statement.

Examination of the Jena installation's vocabulary directory (`/com/hp/hpl/mesa/rdf/jena/vocabulary`), shows several Java classes that wrap Dublin Core, VCARD, and other RDF standards. By using a wrapper class for the properties and resources of a RDF vocabulary, one may define all aspects of the RDF vocabulary in one location, simplifying implementation and maintenance. See the appendix for an example.

APPENDIX B

SPARQL/JENA

Most forms of a SPARQL query contain a set of triple patterns called a *basic graph pattern*. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern *matches* a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is an RDF graph equivalent to the subgraph.

For example, given the RDF triple

```
<http://www.example.com/book/book1>
   <http://purl.org/dc/elements/1.1/title> "Practical RDF"
```

we can execute the query shown below. The title of the book (*Practical RDF*) will be returned as the result.

```
SELECT ?title
WHERE
{
  <http://www.example.com/book/book1>
     <http://purl.org/dc/elements/1.1/title> ?title .
}
```

The result of a query is a *solution sequence*, corresponding to the ways in which the query's graph pattern matches the data. There may be zero, one or multiple solutions to a query.

Consider now an example where multiple values are returned in the solution sequence. The data here is in Turtle format.

Data

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   " Albert Esterline" .
_:a  foaf:mbox   <mailto:esterlin@ncat.edu> .
_:b  foaf:name   "Tim Berners-Lee" .
_:b  foaf:mbox   <mailto: timbl@w3.org> .
 .
```

Query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

Results

Name	**Mbox**
"Albert Esterline"	<mailto:esterlin@ncat.edu>
"Tim Berners-Lee"	<mailto: timbl@w3.org>

Next, consider a working example executing a SPARQL query using the Jena API (data and Java source displayed). A Query object is first created and assigned. This query object contains the string itself representing the SPARQL query.

```
com.hp.hpl.jena.query.Query query =
QueryFactory.create(queryString);
```

After the `Query` construction executes, a `QueryExecutionObject` must be created and assigned. This object will hold the SPARQL text (contained in the `Query` object just created) and will also provide the RDF data itself read in previously.

```
QueryExecution qe = QueryExecutionFactory.create(query, model);
```

The `execSelect()` method of the `QueryExecution` class can now be called, which executes the query and returns the results of the query.

```
ResultSet results =  qe.execSelect();
```

Data (file `info.RDF`)

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

```
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:rss="http://purl.org/rss/1.0/"
        xmlns:dc="http://purl.org/dc/elements/1.1/">

  <foaf:Group>
    <foaf:name>Semantic Web</foaf:name>
    <foaf:homepage rdf:resource=" http://www.w3.org/2001/sw/" />

    <foaf:member>
      <foaf:Person>
        <foaf:name>Albert Esterline</foaf:name>
        <foaf:weblog>
          <foaf:Document rdf:about="http://ncat.edu/~esterlin/">
            <dc:title>Dr. Esterline's Assignments</dc:title>
          </foaf:Document>
        </foaf:weblog>
      </foaf:Person>
    </foaf:member>

    <foaf:member>
      <foaf:Person>
        <foaf:name>Tim Berners-Lee</foaf:name>
        <foaf:weblog>
          <foaf:Document rdf:about="http://www.w3c.org">
            <dc:title>World Wide Web Consortium</dc:title>
          </foaf:Document>
        </foaf:weblog>
      </foaf:Person>
    </foaf:member>

  </foaf:Group>
</rdf:RDF>
```

Java Source

```
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.graph.query.Query;
import com.hp.hpl.jena.ontology.*;
import com.hp.hpl.jena.vocabulary.*;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.util.*;

public class Example3 extends Object {
    public static void main (String args[]) throws IOException {
      InputStream in
```

```
        = new FileInputStream(new File("c:\\jenaroot\\info.rdf"));

      // Create an empty in-memory model, populate it from the graph
      Model model =
             ModelFactory.createMemModelMaker().createDefaultModel();
      model.read(in,null); // null base URI: model URIs are absolute
      in.close();

      // Create a new query
      String queryString =
        "PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
        "SELECT ?url " +
        "WHERE {" +
        "      ?person foaf:name \"Tim Berners-Lee\" . " +
        "      ?person foaf:weblog ?url . " +
        "      }";

      com.hp.hpl.jena.query.Query query =
                                  QueryFactory.create(queryString);

      // Execute the query and obtain results
      QueryExecution qe = QueryExecutionFactory.create(query, model);
      ResultSet results =  qe.execSelect();

      // Output query results
      ResultSetFormatter.out(System.out,
                         (com.hp.hpl.jena.query.ResultSet) results,
                          query);

      // Important - free up resources used running the query
      qe.close();

    }
}
```

When the code is executed on the RDF data `info.RDF`, the results received follows:

```
| url                            |
================================
| <http:// http://www.w3c.org> |
--------------------------------
```

APPENDIX C

PROCESSING OWL WITH JENA

Recall that an ontology model is an extension of the Jena RDF model that provides extra capabilities for handling ontologies. Ontology models are created through the Jena `ModelFactory`. The simplest way to create an ontology model is as follows, where an ontology model is created with default settings:

```
OntModel m = ModelFactory.createOntologyModel();
```

For an extedned example, we use the ontology located at

http://www.eswc2006.org/technologies/ontology#Paper,

Using the Jena ontology API, we create an individual, proceed to display asserted relationships, and, using one of the built-in reasoners, we then display any inferred relationships. Using the ontology mentioned above, the following Java source displays the asserted relationships.

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.ontology.*;
import java.util.*;

public class Example5 extends Object {

  public static void main (String args[]) {

    // create the base model
    String SOURCE = "http://www.eswc2006.org/technologies/ontology";
    String NS = SOURCE + "#";
    OntModel base
              = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM
);
```

```
    base.read( SOURCE, "RDF/XML" );

    OntClass paper = base.getOntClass( NS + "Paper" );
    Individual p1 = base.createIndividual( NS + "paper1", paper );

    for (Iterator i = p1.listRDFTypes(false); i.hasNext(); ) {
      System.out.println( p1.getURI() + " is asserted in class " + i.next()
);
    }
  }
}
```

By including a built-in reasoner with the code,

```
OntModel inf
  = ModelFactory.createOntologyModel(
OntModelSpec.OWL_MEM_MICRO_RULE_INF,
                                        base )
```

we can then produce inferred relationships. The following Java source code displays both the asserted and inferred relationships.

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.ontology.*;
import java.util.*;

public class Example6 extends Object {

  public static void main (String args[]) {

    // create the base model
    String SOURCE = "http://www.eswc2006.org/technologies/ontology";
    String NS = SOURCE + "#";
    OntModel base
          = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM );
    base.read( SOURCE, "RDF/XML" );

    // create the reasoning model using the base
    OntModel inf
      = ModelFactory.createOntologyModel(
                      OntModelSpec.OWL_MEM_MICRO_RULE_INF, base );

    OntClass paper = base.getOntClass( NS + "Paper" );
    Individual p1 = base.createIndividual( NS + "paper1", paper );

    for (Iterator i = p1.listRDFTypes(false); i.hasNext(); ) {
        System.out.println( p1.getURI() + " is asserted in class "
                            + i.next() );
    }

    p1 = inf.getIndividual( NS + "paper1" );
    for (Iterator i = p1.listRDFTypes(false); i.hasNext(); ) {
```

```
        System.out.println( p1.getURI() + " is inferred to be in class
"
                            + i.next() );
    }
  }
}
```