Semantics-Preserving Message Translation

Demonstration of a Time-Space Position Information Example

Reginald Ford, Daniel Elenius, Susanne Riehemann
SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025, USA
reginald.ford@sri.com, daniel.elenius@sri.com, susanne.riehemann@sri.com

October 4, 2011

ABSTRACT
It is common for information exchange between heterogeneous systems to be syntactically correct but fail to achieve its intended purpose due to semantic mismatches. NATO research group IST-094 is developing a framework to facilitate semantic interoperability. We will show a live demonstration of one possible approach to implementing a message translation broker, following the Semantic Interoperability Framework phase structure (i.e., preparatory, configuration, and operational phases). The broker uses a combination of OWL, SWRL, Prolog, Java, XSLT, and Javascript. What follows is an extended abstract of the translation broker demonstration and the introductory presentation that will describe its technical foundations.

1.0 INTRODUCTION

The motivation for and general approach to semantic interoperability that underlies our presentation and demonstration is presented in the "Position Paper on Framework for Semantic Interoperability" (IST-094/RTG-044) [7]. There are many possible ways to realize the framework. In this demonstration we present one that leverages work that SRI has performed for the Open Netcentric Interoperability Standards for Training and Testing (ONISTT) and Analyzer for Netcentric System Test Confederations (ANSC) programs. It also leverages work done in collaboration with other members of the SILF working group. (See Acknowledgements below).

ONISTT and ANSC have developed a conceptual approach for expressing detailed and precise information about systems and their interoperability context, and an automated toolset that applies a reasoning engine to draw conclusions about interaction issues and opportunities (see [1], [2], and [6]). The basic approach is to create knowledge bases that capture formal representations of the capabilities that are needed to do a task, and the capabilities available among candidate resources. We usually focus on tasks that involve interactions between two or more roles. The capabilities needed and capabilities available are weighed against each other by an inference engine to see if there are resources in the candidate pool that have the right kinds of capabilities, and if they are compatible with resources with which they need to interact. We call this purpose-aware interoperability because we're not trying to solve the unbounded question of determining whether two resources are interoperable for any purpose. Although the approach and the toolset were initially developed to facilitate planning for improvisational LVC training and testing events, they are very general in design and can be applied to a wide variety of domains, and all phases of the systems engineering process (e.g., including automated assessment of an as-designed or as-tested system's compliance with its required capabilities).

The ONISTT/ANSC ontologies and Analyzer have been approved for public release. Many of the ontologies are directly applicable to the needs of the Semantic Interoperability Framework. In addition, although semantic translation of information exchanges is not the same as ONISTT/ANSC planning analysis, our inference engine and methods for applying semantic technologies are well suited to provide the core capabilities of a semantic translation broker. This extended abstract describes the ontology context and general technical approach of the broker. It also includes a brief description of the demonstration.

2.0 COMMON MEDIATION RESOURCES

The "Position Paper on Framework for Semantic Interoperability" [7] explains the concept of *Common Mediation Resources*, which includes ontologies. In this section we describe the approach to ontologies underlying our demonstration example.

2.1 Ontology Languages

Ontologies are highly structured and expressive knowledge representation (KR) schemes. Formal logics have been used to capture knowledge since Aristotle. Logics come with *inference rules*, allowing us to draw potentially unanticipated conclusions from asserted facts. The asserted facts could all be very simple, yet the inferred facts can be complex and non-obvious. Because the inference rules are strict if-then rules, their application can be automated. Automated inference is also known as *machine reasoning*, and automated inference systems are usually called *reasoning engines* or *inference engines.*

There are many different logics, e.g., Horn logic, first-order logic, modal logics, higher-order logic, which differ in their expressiveness. More expressive logics come at a price: the automated inference requires more time and computing resources. Description Logics (DL) are a family of logics that represent a good trade-off between expressiveness and tractability for many types of applications.

OWL (Web Ontology Language) is a standardized DL language that has gained wide acceptance in different communities. There are many freely available tools and reasoning engines for OWL. The basic building blocks of OWL are classes, properties, individuals, and axioms that describe how these relate to each other. SWRL (Semantic Web Rule Language) is an extension that makes it possible to write rules expressing mathematical and other relationships that cannot be described in plain OWL.

We use the OWL and SWRL languages for the prototype ontologies used in our demonstration example.

2.2 Ontologies and Knowledge Bases

We can distinguish two types of knowledge acquisition: *building ontologies* and *populating knowledge bases (KBs).* The building of ontologies consists of creating a structure of classes and properties, axioms (restrictions) that relate these to one another, and rules. Population of KBs consists of creating individuals that instantiate the classes in the ontology, and specifying how the individuals are connected to each other.

Building an ontology requires both a deep understanding of the domain and the ability to abstract from specific details to situate specialized knowledge from a sub-field in a larger framework. Once the ontological structure is in place, populating the KB should not require any fundamental thinking. However, the distinction between ontology building and KB population is somewhat blurry – often one discovers flaws in the ontology while populating the initial KB, creating a back-and-forth workflow between the two tasks. Further KBs of a similar type should require fewer or no revisions to the ontology.

Common mediation ontologies range from fundamental scientific or engineering concepts to specific types of systems or resources. Ideally, the ontologies will be developed and maintained by recognized standards bodies or other authoritative organizations or consortia. We do not believe that it is feasible (at least not at present) to develop the "one true ontology" in any domain. However, the unchecked proliferation of ontologies within particular domains is unwise given the difficulty of mapping heterogeneous overlapping ontologies unambiguously and with sufficient detail to support machine reasoning. One approach is to select a set of ontologies as the unifying standard to which other ontologies are mapped instead of directly mapping pairs of ontologies to each other. This is the approach we take in our message translation demonstration. The messages are converted to OWL and are mapped to the unifying ontologies using SWRL.

The OWL ontologies used in the demonstration example include:

- quantity, engineering_value, engineering_measurement
- spatial, iso18026/spatial_reference_frame, nima/wgs84
- time, w3/time_entry
- communication, message_format, protocol
- tspi_projection
- swrl-extensions, swrl-utilities

The information exchange KBs built with reference to these ontologies include:

- nffi13, dis_pdu

Our *engineering_value.owl* is a fundamental ontology that is used extensively in our KBs to represent physical quantities unambiguously. It is common to encounter problems with unstated units of measure, or cases where units are not collocated with the relevant data. *EngineeringValue* subclasses are quantity types like *Mass* or *Length*. *EngineeringValue* individuals contain a *magnitude* and a mandatory *unit*.

Also encoded in the engineering value ontology are conversion factors to the unit of measure in the International System (SI) standard for each particular quantity type. (For example, *meter* is the SI standard unit of length; the conversion factor from *foot* to *meter* is 0.3048.) One or at most two applications of these conversion factors allow conversion between any supported pair of units. This practice is consistent with the approach taken by the National Institute of Standards and Technology (NIST) in [8]. This ontology has been enhanced with SWRL rules that support the definition and usage of quantity *intervals* (e.g., "0 to 100 meters") and the comparison of quantities and quantity intervals.

Quantities are not limited to SI concepts. We have also developed a domain ontology for binary data concepts addressed in IEC 80000-13 [5]. Although such concepts are central to the world of computers, they are inconsistently applied and not always well understood even by industry professionals. Do *you* know the difference between a mebibyte and a megabyte, or why your 700 MB file (actually MiB) won't fit on a 700 MB CD?

Related to the engineering value ontology is *engineering_measurement.owl*. This ontology is based on the practices documented in [3]. *EngineeringMeasurement* individuals are built from an *EngineeringValue* representing the *measurand* and another *EngineeringValue* of the same quantity type representing the *uncertainty* in the measurement (typically one standard deviation). Acknowledging, documenting, and sharing measurement uncertainties can be critical in information exchange about military entities, but they are often unstated or unknown.

Although these quantity ontologies are essential for documenting systems, contents of messages *on the wire* may omit repeated transmission of the same metadata. A simple example is the case where OWL-based, machine-processable metadata declares that all lengths will be exchanged as meters; the unit indicator can then be omitted from the wire message to reduce bandwidth consumption. This approach can also be used to support heterogeneous exchange, for example, one system publishes metadata saying lengths of one class of data it is sending are to be interpreted as meters and lengths of another type as centimeters, while another application declares centimeters and millimeters, respectively.

Another group of related ontologies used in the demonstration example is based on the SEDRIS (Synthetic Environment Data Representation and Interchange Specification) Spatial Reference Model (SRM) standard, which is published as ISO 18026 [4]. SRM ontologies include spatial reference frame (SRF), abstract coordinate system, object reference model, and reference datum. The SRM ontologies are used by the domain KBs describing information exchange standards to identify the spatial reference frames that are associated with their time-space-position information (TSPI) reports. The communication ontologies are used to describe both syntactic and protocol details about how this information is published.

2.3 The SWRL Query Tab and KB Exploration

OWL expressiveness is excellent for constructing rich KBs whose semantics can be understood by a machine, but the KBs can be very difficult for a human to explore with current state-of-the-art tools like Protégé and TopBraid. We have developed a SWRL Query Tab plugin for Protégé that has a panel for entering arbitrary conjunctive OWL/SWRL queries, and then displays the results in tabular format. The results can be saved to a *sparql-results*-formatted XML file, which can then be translated by a simple XSLT script to a suitable display in a language such as HTML.

In our demonstration we will show queries that derive asserted and inferred facts about the example systems. The results will be shown in a browser.

3.0 TECHNICAL APPROACH

3.1 Translation Broker Overview

The common unifying ontologies and KBs used by the translation broker will be created during the SILF *preparation phase*.

In the *configuration phase*, OWL ontologies for specific systems or information exchange models are acquired or created. In our demonstration example, they are auto-generated from XSD descriptions of message syntax and semantics. We customize XSLT and Javascript templates for the particular message formats that specify the series of actions required to orchestrate mediation and translation in the operational phase. In the future we hope to automate this step. We also plan to augment the auto-generated ontologies to improve the mapping to the unifying ontologies.

In the *operational phase*, the broker automatically translates messages. The broker includes Javascript that makes calls to SWRL web services. These services convert an input XML message to OWL, load this OWL data and the relevant OWL ontologies into the reasoner's knowledge base, query the knowledge base, translate the positional data, and apply XSLT to format the output messages. Translation services include direct application of SWRL rules (e.g., to convert a value to SI standard units), and SWRL calls to Java procedural code (e.g., to perform advanced mathematical operations such as coordinate system translation). The SWRL rules are executed by the ONISTT Prolog reasoner.

A SILF operational phase translation broker also needs a realtime capability to acquire and generate on-the-wire or over-the-air messages. We do not anticipate providing this kind of capability in the prototype broker at this time. Instead, message input and output is accomplished via files or pasting into text input boxes.

It is possible that an operational phase broker component such as we are building might be suitable for deployment as a SILF asset in some circumstances, either standalone (if realtime message handling is added), or as component or service employed by a gateway or other mediation application. More generally, the scripts could be used to guide, facilitate, or test the work of the software engineers who are responsible for developing field-deployable systems.

The translation broker described above is complete at the time of this paper, but additional features may be added prior to the IST-097 Workshop demonstration. Section 3.2 describes the essential technical innovations required to implement the broker, none of which is proprietary.

3.2 Technical Discussion

For the main ontologies, OWL is sufficiently expressive. In fact, we make relatively light use of most of its constructs. We do, however, make heavy use of SWRL rules. SWRL allows us to do mathematical computations, and define other complex relationships that are not part of OWL itself. However, we have run into several limitations of SWRL, which we discuss briefly in the following. The first three are explored in more detail in [1].

SWRL requires user-defined predicates to have only one or two arguments because regular OWL classes (unary) and properties (binary) are used as predicates. This effectively limits the language to functions of one variable, because one argument position has to be used for the result of executing the function. There are two ways around this problem, neither of them fully satisfactory. The first is to use an RDF (Resource Description Framework) list to contain several arguments. This makes the rules very verbose and mired down in representation details that make them hard to understand and change. It also means that one cannot "pattern match" on the rule head, which could otherwise make rules more elegant. The second solution is based on the fact that SWRL has so-called built-in predicates which can take an arbitrary number of arguments. In Protégé, we can create our own pseudo-built-ins simply by creating new individuals of the `swrl:Builtin` class. These can be used as n-ary predicates in rules, and fully defined in SWRL itself with no need for an external "built-in" definition. This is the solution that we have adopted, but it is less than ideal because it is unlikely to work in tools and inference engines other than our own.

A second serious limitation of SWRL is the lack of *closed-world reasoning*. OWL and SWRL adopt an *open world assumption*, which means that it is always possible that a fact may be true, even if it is not currently known to be true. The assumption is that we do not know everything there is to know, which is quite reasonable for Semantic Web applications with distributed sources of knowledge. However, it causes difficulties for writing rules, because it is too hard to prove negations under the open-world assumption – we cannot prove that a fact is not true as long as it is at all possible that the fact is true. This is called "classical negation". Some languages, like Prolog, have a "closed-world assumption," and along with it a different form of negation called "negation as failure" (NAF), where a fact is considered to be false if it cannot be *proven* to be true. In other words, one only reasons with locally known information, which makes negations easy to prove. While we cannot entirely drop the open-world assumption, having some form of *local closed-world assumption* is crucial, in our experience. For now we have adopted the following solution: we have added a new built-in predicate `allKnown` which returns a list of all the *known* values for a given individual and property. A list is inherently a closed-off collection on which we can conveniently perform various kinds of computations. Again, this is not an ideal solution – reasoning engines have to implement this new built-in for our rules to work (of course, our own engine is currently the only one that does implement it), and a more general form of local closed-world reasoning would be preferable.

A third limitation of SWRL is its inability to *create new individuals* as a result of evaluating rules. SWRL can create new relationships between individuals, or calculate numbers, but cannot create new individuals. This limitation initially motivated us to consider SPARQL and SPIN as an alternative to SWRL. SPARQL *CONSTRUCT* can create new individuals, and SPIN can associate SPARQL queries with specific OWL classes. However, we found the gain to be inadequate compensation for the losses. SPARQL only looks at asserted RDF triples instead of inferencing based on OWL language axioms, which means that reasoning using SPARQL in effect requires re-implementing semantics that are native to OWL and SWRL. In addition, we find the procedural character of SPARQL queries to be much more verbose and difficult to work with than the declarative elegance of SWRL rules. The technical solution described in Section 3.0 does not require creating new OWL individuals, and thus this limitation of SWRL is not relevant to the design of the translation broker.

Although we have succeeded in implementing fairly complex mathematical functions such as matrix operations in SWRL, there are many advanced functions (e.g., differential equations) that are not feasible. In addition, it is inadvisable to try to reimplement some well-tested and trusted code. SEDRIS coordinate transformations are an example to which both reasons apply. Recently, we have combined the declarative benefits of OWL and SWRL, the reasoning ability of our Prolog inference engine, and the unrestricted functional capability of procedural code, by creating a general capability to call Java procedural attachments from SWRL.

4.0 EXAMPLE

The demonstration example uses the configuration phase and operational phase components that are described in Section 3.1 to translate TSPI (time-space position) information from NFFI (NATO Forces Friendly Information) messages to DIS (Distributed Interactive Simulation) PDUs (protocol data units). The semantic distance between corresponding elements of the two information exchange standards (e.g., coordinate systems, timestamps) is considerable. This example represents a continuation of semantic analysis and ontology development to which the SILF team contributed during the November 2010 and March 2011 meetings.

What follows is a notional description of the demonstration. Exact details and sequencing of the final version of the November presentation may differ.

The demonstration will begin with an end-to-end functional view of the operational phase broker, including:

- The positional data fields of an XML source NFFI message report are shown. A SWRL query invokes a Java attachment to display the position on a Google map.

- The NFFI message is presented to the Javascript broker. The broker first converts it to OWL and loads this OWL data and the relevant OWL ontologies into the reasoner's knowledge base. It then invokes SWRL Query endpoint services to query the knowledge base and to translate the positional data. The translation includes unit conversions, timestamp conversion, and a Java attachment to perform SEDRIS geodetic-to-geocentric coordinate transformations. Finally it applies XSLT to transform the query results into a DIS message in XML format.

- The fields of the output DIS message are compared with the source NFFI message.

We will then show and discuss the constituent parts of the ontologies, configuration phase broker, and operational phase broker, including:

- The auto-generated NFFI and DIS ontologies.
- The unifying spatial and time ontologies.
- The SWRL rules that convert NFFI to DIS.
- The XSLT that generates the destination DIS/NFFI messages.
- The customized Javascript template that controls calls to SWRL web services that perform the NFFI-to-DIS and DIS-to-NFFI translations.
- Description of the Prolog reasoner that processes the SWRL rules.
- Description of the Java software that creates a SWRL Query endpoint.

Some protocol differences between the two message standards (e.g., the expectations for conditions controlling the timing and frequency of position updates) are not addressed in this translation. The demonstration may include application of the ONISTT analyzer to:

- Issue a warning message that NFFI does not implement DIS *dead reckoning (DR)* algorithms, and that NFFI systems may not receive updates from DIS systems at the expected rate.
- If information is available regarding position report update rate expectations of systems using NFFI, a *configuration artifact* specifying the *heartbeat* (i.e., maximum update interval) to which participating DIS systems should be configured.
- Issue a warning that some position report timestamps may be unreliable due to their one-hour rollover interval.
- After adding a DIS protocol adapter (e.g., Simulation C2 Interchange Module for Plans, Logistics, and Exercise) to the pool of resources available for use in the operation, the analysis is rerun and no warning is issued. (The adapter acts as a surrogate for destination systems that do not implement the DIS DR protocol. It outputs dead reckoned positions in accordance with DR error threshold configuration agreements. It also prevents timestamp anomalies that may occur at the boundaries of a rollover interval.)

ONISTT warnings and configuration artifacts could be provided as metadata to systems participating in the operation.

ACKNOWLEDGEMENTS

The work described in this paper was carried out at the SRI facilities in Menlo Park, CA and was funded as part of the ONISTT project by ODUSD/R/RTPP (Training Transformation) under Prime Contract W15P7T-06-D-E402, Subcontract agreement S10-106942. Portions of the ONISTT technology described herein were funded by the U.S. Department of Defense, TRMC (Test Resource Management Center) T&E/S&T (Test and Evaluation/Science and Technology) Program under NST Test Technology Area prime contract N68936-07-C-0013.

We would also like to thank Bjrn Jervell Hansen and Dariusz Nogalski for their expert work in developing the NFFI-related ontologies employed in the example.

REFERENCES

[1] D. Elenius, D. Martin, R. Ford, and G. Denker. Reasoning about Resources and Hierarchical Tasks Using OWL and SWRL. In *International Semantic Web Conference*, volume 5823 of *Lecture Notes in Computer Science*, pages 795–810. Springer, 2009.

[2] R. Ford, D. Martin, D. Elenius, and M. Johnson. Ontologies and tools for analyzing and composing simulation confederations for the training and testing domains. *Journal of Simulation*, Special Issue: Enhancing Simulation Composability and Interoperability using Conceptual/Semantic/Ontological Models, Andreas Tolk and John A. Miller, eds., August 2011.

[3] ISO, Geneva. *Guide to the expression of uncertainty in measurement*, 1995.

[4] ISO, Geneva. *Information technology - Spatial Reference Model (SRM)*, 2006.

[5] ISO, Geneva. *Quantities and units Part 13: Information science and technology*, 2008.

[6] S. Riehemann and D. Elenius. Ontological analysis of terrain data. In *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, COM.Geo '11, pages 10:1–10:8, New York, NY, USA, 2011. ACM.

[7] SILF working group. Position paper on framework for semantic interoperability. Draft, IST-094/RTG-044, 2011.

[8] A. Thompson and B. N. Taylor. Guide for the use of the international system of units (SI). Technical Report Special Publication 811, National Institute of Standards and Technology, Gaithersburg, MD, 2008.