

# TRANSLATOR: A TRANSLator from LAnguage TO Rules\*

David Z. Hirtle

Faculty of Computer Science  
University of New Brunswick  
PO Box 4400  
Fredericton, N.B. E3B 5A3, CANADA  
david.hirtle@unb.ca

## Abstract

One explanation as to why the Semantic Web has not quite caught on yet is that the barrier to entry is too high. This paper describes TRANSLATOR, a free tool available as a Java Web Start application designed to allow anyone, even non-experts, to write facts and rules in formal representation for use on the Semantic Web. This is accomplished by automatically translating natural language sentences written in Attempto Controlled English into the Rule Markup Language, using the Attempto Parsing Engine webservice as a backend. XML representation in RuleML has several advantages, not least of which is compatibility with existing Semantic Web standards.

## 1 Introduction

The Semantic Web, often described as the next generation of the existing Web, is a highly collaborative effort with the ultimate goal of making it easier for humans and computers to work together [3]. The key is enriching the current Web, which is understandable by humans alone, with machine-interpretable (meta)data in the form of facts, rules and ontologies.

Enabling standards such as the Resource Description Framework (RDF) and Web Ontology Language (OWL) have already been defined by the World Wide Web Consortium (W3C), and work toward a Rule Interchange Format (RIF) is now in progress. Despite all of this activity, the Semantic Web has yet to see widespread adoption. There is some concern (e.g., [10]) that the barrier to entry is too high: understanding such formal languages is a challenge for the vast majority of individuals.

One way of lowering the barrier to entry is to offer a more user-friendly format. While the intuitiveness and expressiveness of natural language makes it a logical candidate, it is inherently

---

\*Special thanks to Harold Boley for supervising the honours thesis upon which this paper is based.

ambiguous and has therefore largely been ignored for formal applications. Fortunately, recent work has shown that principles from computational linguistics can be applied to translate between language and machine-interpretable logic. This is accomplished by using a subset of language that remains completely natural yet lends itself to formalization. The most mature such domain-independent *controlled* language is Attempto Controlled English (ACE) [6], whose development over the past ten years has also included its own Attempto Parsing Engine (APE) [8].

The focus of this work is using ACE as the means to author formal rules for the Semantic Web via a newly-developed application called TRANSLATOR (TRANSlator from LAnguage TO Rules). Specifically, rules written in ACE are taken as input, parsed (using the APE webservice) into a variant of first-order logic known as a Discourse Representation Structure (DRS) and then mapped into formal representation in the XML-based Rule Markup Language (RuleML) [4]. RuleML is the product of a long-standing effort toward standardizing rule markup on the Web and is compatible with other Semantic Web languages.

TRANSLATOR, available as a Java Web Start application<sup>1</sup>, enables non-programmers to write their own rules on the Semantic Web via a subset of English that remains completely familiar. As a first example, users may write:

*If the customer is a student then he/she receives a discount.*

instead of something formal like the following (the corresponding DRS in logical notation):

$$\begin{aligned} &\forall ABC : \text{object}(A, \text{atomic}, \text{student}, \text{person}, \text{cardinality}, \text{count\_unit}, \text{eq}, 1) \wedge \\ &\text{predicate}(B, \text{state}, \text{be}, C, A) \wedge \\ &\text{object}(C, \text{atomic}, \text{customer}, \text{person}, \text{cardinality}, \text{count\_unit}, \text{eq}, 1) \\ &\rightarrow \\ &\exists DE : \text{object}(D, \text{atomic}, \text{discount}, \text{object}, \text{cardinality}, \text{count\_unit}, \text{eq}, 1) \wedge \\ &\text{predicate}(E, \text{unspecified}, \text{receive}, A, D) \end{aligned}$$

## 2 Input: Attempto Controlled English

The input accepted by TRANSLATOR has the appearance of plain English but is in fact a formal language in the same vein as the relatively esoteric RDF and RDF-based OWL. Known as Attempto Controlled English (ACE), it is a tractable subset of full English that can be unambiguously translated into first-order logic. In this way, ACE combines the ease of use and familiarity of natural language with the ease of processing and precision of formal language.

### 2.1 Overview

The language of ACE remains as natural as possible while avoiding ambiguity. To this end, the grammar of ACE excludes certain imprecise phrasings, e.g.,

*Students hate annoying professors.*

(Do students hate professors who are annoying, or do they hate to annoy professors?) Remaining ambiguity is handled by predefined rules favouring one interpretation over others. Consider the following sentence:

---

<sup>1</sup><http://www.ruleml.org/translator>

*The student brings a friend who is an alumnus and receives a discount.*

This is ambiguous in full natural language: does the student or the friend receive the discount? In ACE, on the other hand, the sentence has the interpretation that it is the student who receives the discount, i.e. the coordination is outside the relative phrase. The alternative interpretation (the friend receives the discount) can be expressed in ACE by simply repeating the relative pronoun:

*The student brings a friend who is an alumnus and **who** receives a discount.*

The vocabulary of ACE consists of predefined and user-defined elements. Function words such as articles (e.g., *a, the*), prepositions (e.g., *in, to*) and pronouns (e.g., *they, itself, everyone*) as well as fixed phrases (e.g., *it is not the case that...*) are all predefined. On top of that, a default lexicon of nearly 100 000 content words (nouns, verbs, adjectives and adverbs) can be supplemented by user-defined (e.g., domain-specific) lexicons.

This brief overview of ACE has not covered several significant features such as anaphoric references and plurals. Complete details on ACE, including further publications, are available from the Attempto project website<sup>2</sup>.

## 2.2 Expressible Rules

While all ACE sentences are accepted English, the opposite is not true. Since TRANSLATOR focuses on translating language (i.e., ACE) to rules, a pertinent question becomes, “What kinds of rules are expressible in ACE?”

First, the notion of a rule is introduced. Knowledge is divisible into two categories: facts (e.g., *John is human*) and rules, which allow inferring new facts from existing ones (e.g., *All humans are mortal*, giving the new fact *John is mortal*). The typical logic programming (e.g., Prolog) representation of a rule is as a left-hand side (i.e., conclusion) separated from a right-hand side (i.e., one or more conditions) by a special “is implied” symbol, e.g.,

`mortal(X) :- human(X).`

This can be translated to English as *if X is human then X is mortal*. The representation of a fact has only a left-hand side, e.g.,

`human(John).`

In other words, *John is human*. Since there are no conditions, the conclusion is unconditionally true. Therefore facts are just a special kind of rule and rules can be considered as conditional facts.

It is important to realize that rules are expressible in natural language in a variety of (possibly ambiguous) ways. Indeed, the same general rule can be expressed in many syntactically different forms, e.g.,

*Everyone is mortal.*

*All humanity is mortal.*

*Every human being is mortal.*

*For each person the person is mortal.*

*If there is a member of the human race then he/she is mortal.*

---

<sup>2</sup><http://www.ifi.unizh.ch/attempto>

All of the above rules are, in fact, valid ACE. Furthermore, each such pattern of rule can be further embellished with negation, relative clauses, etc. For example:

*Every honest student who does not procrastinate receives a good mark and always passes the course easily.*

TRANSLATOR also adds support for “infix” implication, e.g.,

*The student is happy **if** there is no class.*

The conclusions on the right-hand side are swapped with the conditions on the left-hand side and the word *then* is inserted, yielding the following equivalent (but now valid ACE) version:

*If there is no class then the student is happy.*

This preprocessing step is done before the input is sent to the APE webservice so that it never sees the original version.

There are actually several different kinds of rules; the examples so far have been derivation rules (also known as inference or deductive rules), which are primarily used for reasoning applications and do not invoke any actions. This category of rules is the focus for TRANSLATOR since RuleML is already well-developed in this area and the DRSs output by APE directly support this kind of implication.

## 2.3 Discourse Representation Structures

The parsing engine APE, accessed as a webservice by TRANSLATOR, translates ACE texts into a syntactic variant of first-order logic called a Discourse Representation Structure (DRS). The utility of DRSs is a result of Discourse Representation Theory, a formal method of dealing with contextual meaning across multiple sentences [12]. In particular, DRSs provide a solution to the problem of anaphora resolution, e.g., whether *it* refers to the cat or the dog below:

*The owner separates the cat from the dog. It growls.*

While only a brief introduction is given here, complete details are available in an Attempto report dedicated to DRSs [7].

A single DRS represents an entire discourse (group of sentences). Each entity (e.g., noun, verb) of the discourse is assigned a discourse referent (variable), which is listed at the top of the DRS. Below the referents is a list of related conditions (logical atoms). For example, a DRS for the discourse above might have referents *D*, *B* and *C* and five conditions, conventionally formatted as a box:

<i>D B C</i>
<i>owner(D)</i> <i>cat(B)</i> <i>dog(C)</i> <i>separate(D,B,C)</i> <i>growl(C)</i>

Notice that it is the dog *C* that growls—the anaphor has been resolved.

In the Attempto system, the notation is somewhat different. Most importantly, predefined condition names are used so that, e.g., *cat*(*B*) becomes *object*(*B*, *atomic*, *cat*, *object*, *cardinality*, *count\_unit*, *eq*, 1):

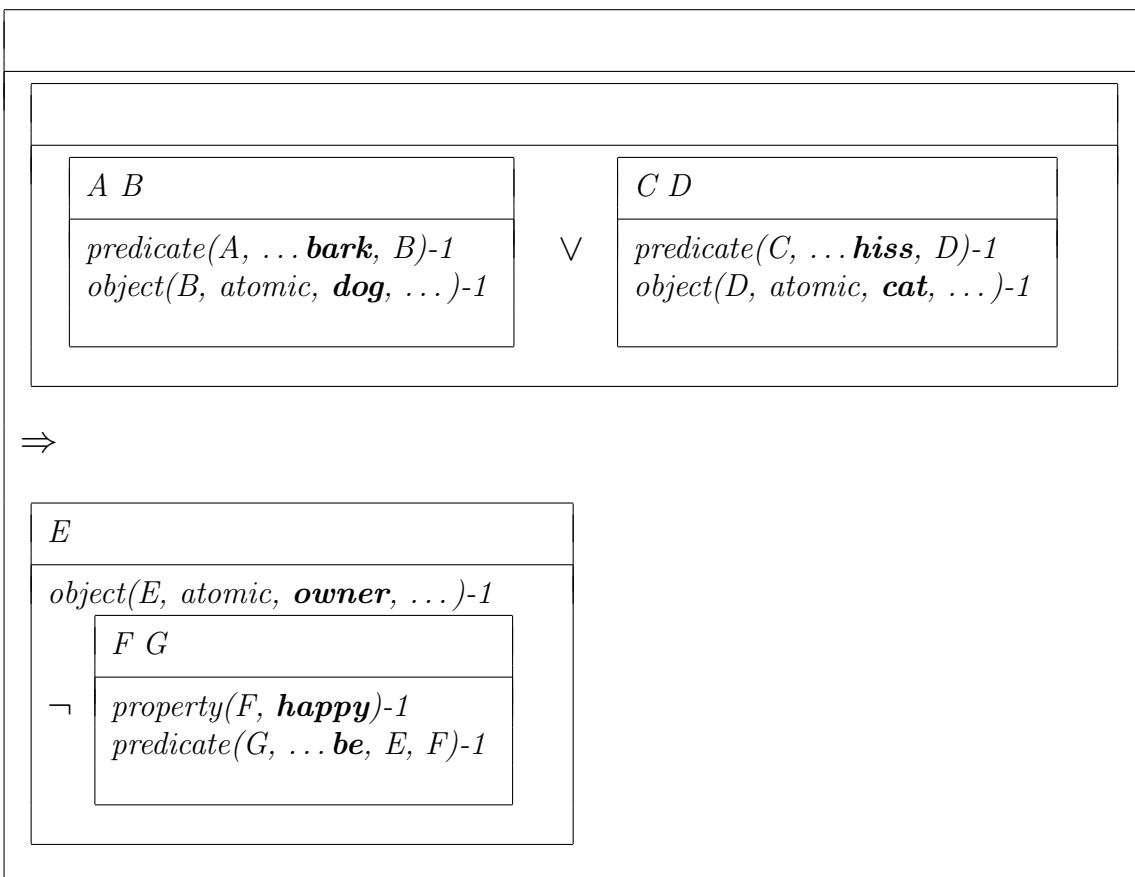
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
				<i>predicate</i> ( <i>A</i> , <i>unspecified</i> , <b><i>separate</i></b> , <i>D</i> , <i>B</i> )-1
				<i>modifier</i> ( <i>A</i> , <i>unspecified</i> , <i>from</i> , <i>C</i> )-1
				<i>object</i> ( <i>D</i> , <i>atomic</i> , <b><i>owner</i></b> , <i>person</i> , <i>cardinality</i> , <i>count_unit</i> , <i>eq</i> , 1)-1
				<i>object</i> ( <i>B</i> , <i>atomic</i> , <b><i>cat</i></b> , <i>object</i> , <i>cardinality</i> , <i>count_unit</i> , <i>eq</i> , 1)-1
				<i>object</i> ( <i>C</i> , <i>atomic</i> , <b><i>dog</i></b> , <i>object</i> , <i>cardinality</i> , <i>count_unit</i> , <i>eq</i> , 1)-1
				<i>predicate</i> ( <i>E</i> , <i>unspecified</i> , <b><i>growl</i></b> , <i>C</i> )-2

This “reified” representation allows using first-order logic where higher-order would normally be required. The Attempto representation also clearly introduces additional referents and conditions to provide additional detail (e.g., quantity information). The integer at the end of each condition is an index indicating the number of the corresponding sentence within the discourse.

The conditions so far have all been simple, but complex ones can be formed with nested DRSs and logical implication ( $\Rightarrow$ ), negation ( $\neg$ ) and disjunction ( $\vee$ ). For example, the ACE text

*If the dog barks or the cat hisses then the owner is not happy.*

is translated into the following DRS:



TRANSLATOR directly translates such DRSs into RuleML. The DRS-RuleML mapping is described in section 4.3.

## 3 Translation

Translating English into RuleML is complicated. Fortunately, the APE webservice greatly simplifies the task. Since details on the Attempto parser are already published [8], the focus here is on the user interface of TRANSLATOR and the series of steps that comprise the translation process.

### 3.1 User Interface

The two primary design goals of TRANSLATOR were that it be

- widely accessible, and
- user friendly.

In fact, both goals are satisfied by its availability as a Java Web Start application: TRANSLATOR can be accessed (cross-platform) online and provides users with an easy to use graphical user interface depicted in Figure 1.

The user interface is divided into an input pane on the top and an output pane on the bottom. The input pane contains a scrolling text area where the user may enter an ACE text as well as the translation button. While translation is in progress (e.g., the APE webservice is being accessed), the translation button is temporarily disabled. The interface is multi-threaded in order to remain responsive to the user during this time.

The output pane consists of four checkboxes, a non-editable scrolling text area where the translation results are displayed, a help button and a save button. The checkboxes allow the user to customize what output is displayed in addition to the RuleML (e.g., the original ACE text, DRS, etc). The help button directs the user's browser to the TRANSLATOR webpage where additional information is available. Finally, the save button will save the current output to a file on the user's local machine for convenience.

### 3.2 Procedure

The translation process can be broken down into the following eight steps:

1. Accept the ACE text input by the user.
2. Preprocess the user input:
  - Reject any queries (ending with “?”) since they are currently unsupported.
  - Add final “.” if omitted. (Valid ACE sentences end with a period.)
  - If “infix implication” found, rearrange `Y if X` into `If X then Y`.
3. Encode the input in a querystring and use it to access the APE webservice.

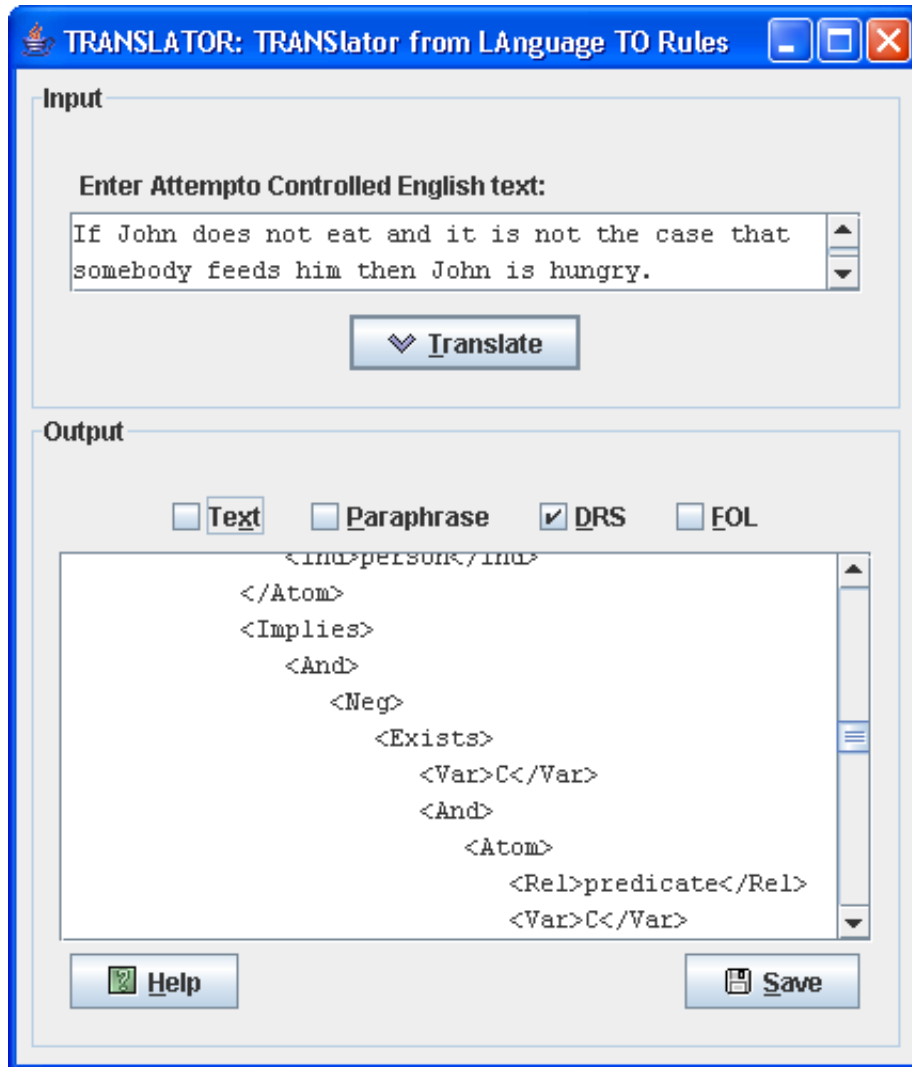


Figure 1: The graphical user interface of TRANSLATOR.

4. Retrieve XML-wrapped result from APE. (This may take several seconds.)
5. Parse result to get messages (if any), DRS, paraphrase, etc.
6. Display messages from APE, if any. End here if the result is otherwise empty (i.e. input was invalid ACE).
7. Traverse the DRS to build RuleML/XML.
8. Display the pretty-printed RuleML (and optional user-configurable extras) to the user.

## 4 Output: Rule Markup Language

The goal of the international Rule Markup Initiative is to define a canonical language for publishing and sharing rules on the Web. The result is the XML-based Rule Markup Language (RuleML), used for derivation, query, transformation, integrity checking and reactive behavior. RuleML is implemented with XML Schema, XSL Transformations (XSLT) and reasoning engines [2]. The RuleML Initiative collaborates with several standards bodies including W3C, OMG and OASIS.

### 4.1 Advantages

First of all, why RuleML as the target formalism? There are, in fact, several advantages to representation in RuleML.

RuleML inherits some of its benefits directly from XML, including platform independence and interoperability. Exchange is simplified because conformance to the RuleML specification (defined in XML Schema) can be confirmed using a validator such as the W3C's XSV. RuleML is also extensible, a prime example being its combination with OWL to form the Semantic Web Rule Language (SWRL) [11]. RuleML is also translatable to and from other Semantic Web standards (e.g. RDF, OWL and probably the upcoming RIF) via XSLT. Various tools are also available, including OO jDREW [1], Mandarax<sup>3</sup>, and NxBRE<sup>4</sup>. Another useful tool is a bidirectional converter between RuleML and the more compact positional-slotted (POSL) syntax combining Prolog and F-logic [5].

### 4.2 Modularization

The Rule Markup Language is actually a family of sublanguages realized with modular XML Schemas, following the general software engineering principle of modularity. A UML-like model of the modularization<sup>5</sup> of RuleML is shown in Figure 2.

Each sublanguage, represented as an unshaded rectangle, corresponds to a well-known rule system (e.g., datalog and hornlog), allowing users to pick and choose according to their specific needs. TRANSLATOR, for instance, uses the first-order logic sublanguage “folog”. The shaded rectangles in the figure represent groups of related sublanguages.

---

<sup>3</sup><http://mandarax.sourceforge.net>

<sup>4</sup><http://www.agilepartner.net/oss/nxbre>

<sup>5</sup>For the complete picture, see <http://www.ruleml.org/modularization>.

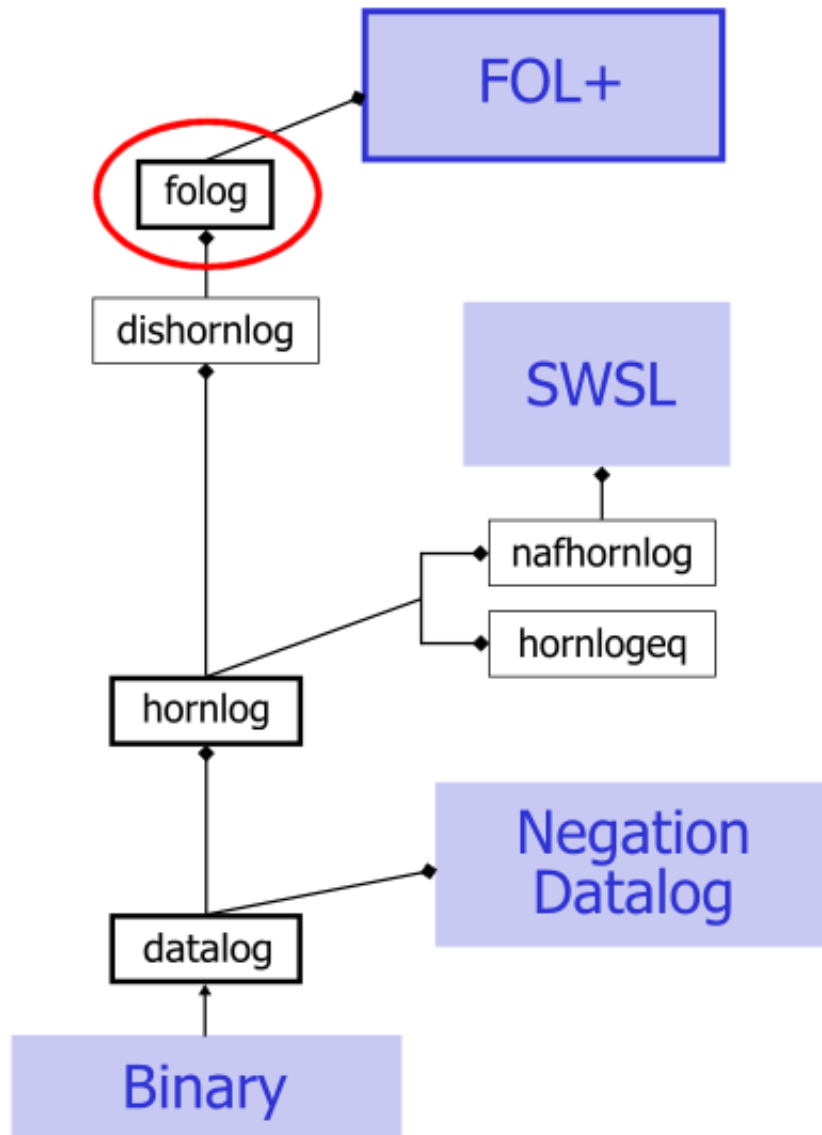


Figure 2: The modularization of RuleML.

### 4.3 DRS-RuleML Mapping

There are many possible mappings between DRSs and RuleML. An important criterion when selecting the mapping was reversibility in order to accommodate the future possibility of translation from RuleML back to ACE. Therefore, it is important that there be no loss of information; in particular, all details of the extended DRS notation used in the Attempto system should be preserved.

A direct mapping is therefore best. For example, the ACE text

*The professor is mortal.*

with the DRS

$A \ B \ C$
$property(A, \mathbf{mortal})-1$ $predicate(B, \textit{state}, \mathbf{be}, C, A)-1$ $object(C, \textit{atomic}, \mathbf{professor}, \textit{person}, \textit{cardinality}, \textit{count\_unit}, \textit{eq}, 1)-1$

is straightforwardly mapped into RuleML as follows:

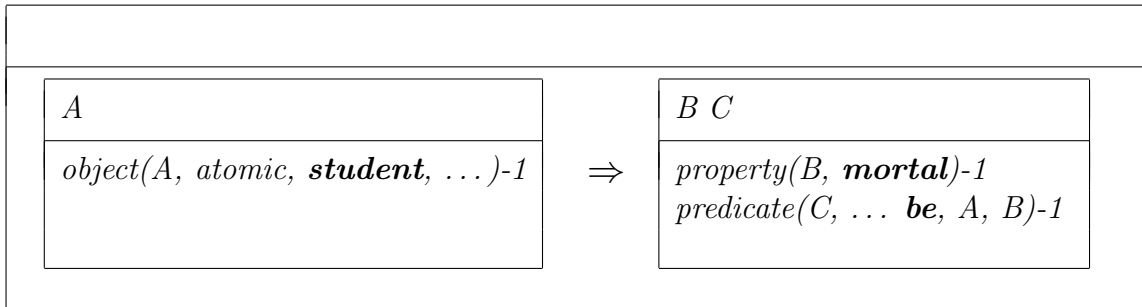
```
<RuleML ... xmlns="http://www.ruleml.org/0.9/xsd">
  <Assert>
    <Exists>
      <Var>A</Var><Var>B</Var><Var>C</Var>
      <And>
        <Atom><Rel>property</Rel><Var>A</Var><Ind>mortal</Ind></Atom>
        <Atom>
          <Rel>predicate</Rel>...<Ind>be</Ind><Var>C</Var><Var>A</Var>
        </Atom>
        <Atom><Rel>object</Rel><Var>C</Var>...<Ind>professor</Ind>...</Atom>
      </And>
    </Exists>
  </Assert>
</RuleML>
```

Finally, the quantification of variables is made explicit in RuleML for clarity. It is implicit in a DRS that all variables are existentially quantified; RuleML has the `<Exists>` tag for this purpose. The only exceptions are variables on the condition side of an implication, which are universally quantified. In RuleML, these are placed within a `<Forall>` tag.

As a concrete example, the ACE text

*Every student is mortal.*

results in the DRS



whose quantification is made explicit in the mapping to RuleML as follows (where the quantifier tags are boxed for clarity):

```

<RuleML ... xmlns="http://www.ruleml.org/0.9/xsd">
  <Assert>
    <Forall>
      <Var>A</Var>
      <Implies>
        <And>
          <Atom><Rel>object</Rel><Var>A</Var>...<Ind>student</Ind>...</Atom>
        </And>
        <Exists>
          <Var>B</Var><Var>C</Var>
          <And>
            <Atom><Rel>property</Rel><Var>B</Var><Ind>mortal</Ind></Atom>
            <Atom>
              <Rel>predicate</Rel>...<Ind>be</Ind><Var>A</Var><Var>B</Var>
            </Atom>
          </And>
        </Exists>
      </Implies>
    </Forall>
  </Assert>
</RuleML>

```

A more standard non-reified representation in RuleML (e.g., with nouns and verbs as relations) could be derived from the version above using advanced XSLT translation. One possibility is as follows:

```

<RuleML ... xmlns="http://www.ruleml.org/0.9/xsd">
  <Assert>
    <Implies>
      <Atom><Rel>student</Rel><Var>A</Var></Atom>
      <Atom><Rel>mortal</Rel><Var>A</Var></Atom>
    </Implies>
  </Assert>
</RuleML>

```

Note that the explicit quantification and conjunctions are gone, and that the atoms have been un-reified (to have more meaningful relation names). Finally, the following atom has been replaced by variable equality:

<Atom><Rel>predicate</Rel>...<Ind>be</Ind><Var>A</Var><Var>B</Var></Atom>

However, this extra measure is only necessary for irregular verbs (such as “to be”).

## 5 Conclusions and Future Work

TRANSLATOR is an open source tool that allows anyone to write facts and rules in formal representation for use on the Semantic Web. It accomplishes this by taking input written in ACE, accessing the APE webservice and then translating the generated DRS into RuleML. The hope is that this user-friendly front-end will help lower the barrier to entry of the Semantic Web and encourage non-experts to get involved; this seems to have been a critical factor in the success of the original Web.

One potential use of TRANSLATOR is to construct real-world use cases, e.g., for company policies. An open issue is that the output of TRANSLATOR, as first-order logic, is undecidable and therefore not supported by reasoning engines. However, reorganization into a decidable subset should be possible, e.g., by factoring out multiple conclusions of a rule.

There are several possible areas of future work. First, TRANSLATOR does not currently support queries because currently the DRS for

*Does the man enter a card?*

is identical to the DRS for

*The man enters a card.*

and it is impossible to identify when to use <Query> versus <Assert> in RuleML. Another possibility is making TRANSLATOR bidirectional by supporting translation of RuleML back into ACE. If the prototype DRS-to-ACE verbalization tool DRACE [9] were accessible (e.g., as another webservice), extending TRANSLATOR to be bidirectional would simply be a matter of reversing the DRS-RuleML mapping. The Attempto team is also continually extending ACE; in fact, the next version includes support for the passive voice and modality, e.g., possibility (*can*) and necessity (*must*). TRANSLATOR should, of course, be updated to support such extensions.

Meanwhile, development of the RIF standard continues. Fortunately, the future challenge of relating TRANSLATOR to RIF (once defined) should be straightforward since RuleML will most likely be translatable to RIF. TRANSLATOR would therefore be one such tool alluded to in the RIF Working Group charter<sup>6</sup>, which states: “Users are expected to work with tools or rule languages which are transformed to and from this format.”

TRANSLATOR is available as a Java Web Start application at <http://www.ruleml.org/translator> and is also linked from the Attempto website. Further collaboration with the Attempto team is anticipated.

---

<sup>6</sup><http://www.w3.org/2005/rules/wg/charter>

## References

- [1] Marcel Ball. OO jDREW: Design and Implementation of a Reasoning Engine for the Semantic Web. Honours Thesis, 2005. <http://www.jdrew.org/oojdrew/docs.html>.
- [2] Marcel Ball, Harold Boley, David Hirtle, Jing Mei, and Bruce Spencer. Implementing RuleML Using Schemas, Translators, and Bidirectional Interpreters. W3C Workshop on Rule Languages for Interoperability Position Paper, April 2005. <http://www.w3.org/2004/12/rules-ws/paper/49>.
- [3] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
- [4] Harold Boley. Object-Oriented RuleML: User-Level Roles, URI-Grounded Clauses, and Order-Sorted Terms. In *RuleML*, pages 1–16, 2003.
- [5] Harold Boley. POSL: An Integrated Positional-Slotted Language for Semantic Web Knowledge, May 2004. <http://www.ruleml.org/submission/ruleml-shortation.html>.
- [6] Norbert E. Fuchs, Stefan Höfler, Kaarel Kaljurand, Fabio Rinaldi, and Gerold Schneider. Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines. In *Reasoning Web*, pages 213–250, 2005.
- [7] Norbert E. Fuchs, Stefan Höfler, Gerold Schneider, and Uta Schwertel. Extended Discourse Representation Structures in Attempto Controlled English. Technical Report ifi-2005.08, Department of Informatics, University of Zurich, 2005.
- [8] Norbert E. Fuchs, Kaarel Kaljurand, Fabio Rinaldi, and Gerold Schneider. Deliverable I2-D3. A Parser for Attempto Controlled English. Technical report, REWERSE, 2005. <http://rewerse.net/deliverables.html>.
- [9] Norbert E. Fuchs, Kaarel Kaljurand, and Gerold Schneider. Deliverable I2-D5. Verbalising Formal Languages in Attempto Controlled English I. Technical report, REWERSE, 2005. <http://rewerse.net/deliverables.html>.
- [10] Stefan Haustein and Jörg Pleumann. Is Participation in the Semantic Web Too Difficult? In *International Semantic Web Conference*, pages 448–453, 2002.
- [11] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosf, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, May 2004. <http://www.w3.org/Submission/SWRL>.
- [12] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht, 1993.