TRANSLATOR: A TRANSLator from LAnguage TO Rules

David Hirthle

David R. Cheriton School of Computer Science
University of Waterloo

(Work done at the University of New Brunswick)

October 13, 2006
Outline

• Introduction
• Translating Language to Rules
• Attempto Controlled English
• Discourse Representation Structures
• Rule Markup Language
• Future Work
• Conclusion
Introduction

• Semantic Web still not widely used
  – Focus: machine-readable (meta)data
    • facts, rules and ontologies
  – Problem: only experts can contribute
    • formal standards like RDF and OWL (and soon RIF) are difficult to learn
    • need to lower the barrier to entry
Example: Semantic Web rule

Every student gets a discount of 15 percent.

```xml
<Implies>
  <body>
    <Atom>
      <Rel>student</Rel>
      <Var>customer</Var>
    </Atom>
  </body>
  <head>
    <Atom>
      <Rel>discount</Rel>
      <Var>customer</Var>
      <Data>15%</Data>
    </Atom>
  </head>
</Implies>
```
Our Approach

• Provide a user-friendly format
  – why not English?
    • easy
    • familiar
    • expressive
    • but ambiguous
  – “controlled English” avoids ambiguity
    • formal, yet natural
Every student gets a discount of 15 percent.

Java Web Start
Step 1: Input

![Diagram showing the flow of input from ACE to GUI, DRS, and DRS Parser, with interactions involving RuleML, APE webservice, and JavaCCGRAMMAR.](image-url)
• Looks like English
  – *Every honest student who does not procrastinate receives a good mark and easily passes the course.*

• Actually a formal language, like RDF
  – tractable subset of English
    • all ACE sentences are English, but not vice versa
  – unambiguously translatable into logic
• Strategies for handling ambiguity:
  – exclude imprecise phrasings
    • Students hate annoying professors.
  – interpretation rules
    • The student brings a friend who is an alumnus and receives a discount.
      – Who receives the discount?
        » in ACE, student does (by default)
        » repeat relative pronoun for other interpretation
          The student brings a friend who is an alumnus and who receives a discount.
How can rules be expressed?

- in natural language, many 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.*

- all above are valid ACE
- further embellishment
  - negation, relative clauses, etc.
• What can’t yet be easily expressed?
  – “infix” implication
    • The student is happy if there is no class.
    • but TRANSLATOR supports it
      – just swap condition(s) and conclusion(s)
        » result: If there is no class then the student is happy.

  – production and reaction rules
    • involve actions
      – If a student is caught cheating then send a report to the registrar’s office.
    • require imperative mood (not yet in ACE)
Step 2: Query APE for DRS
Discourse Representation Structures

- Output by Attempto Parsing Engine (APE)
- Syntactic variant of first-order logic
  - facilitates translation to RuleML
- Basis is Discourse Representation Theory
  - formal way to handle contextual meaning across multiple sentences
  - developed by Hans Kamp (1981)
- APE uses extended “flat” notation
  - e.g., student(X) \(\rightarrow\) object(X,...,student,...)
Every honest student who does not procrastinate receives a good mark and easily passes the course.
Step 3: Parse DRS

ACE
RuleML, ...

query
(ACE + params)

APE webservice

DRS, ...

JavaCC

DRS Grammar

DRS Parser

GUI

RuleML

DRS grammar

TRANSLATOR
Step 4: Map to RuleML

- ACE
- RuleML, ...

GUI
- query (ACE + params)
- DRS, ...

APE webservice

DRS Parser
- DRS
- RuleML

TRANSLATOR

JavaCC

DRS grammar
DRS-to-RuleML Mapping

• Performed “on-the-fly” by actions (Java code) embedded in DRS grammar

• **Direct**
  – preserves extended notation
  – uses positional RuleML syntax

• **Explicit**
  – e.g., quantifiers: `<Forall>`, `<Exists>`

• **Reversible**
  – enables future rules → English extension
Every honest student who does not procrastinate receives a good mark and easily passes the course.

(DRS)

(RuleML)
Rule Markup Language (1)

• Goal is interoperable rule markup
  – XSLT translators to other Semantic Web languages
• Family of “sublanguages”
  – modular XML Schemas
  – each represents well-known rule system
  – TRANSLATOR uses First-Order Logic sublanguage
Rule Markup Language (2)

• Why use RuleML?
  – ease of interchange (XML)
  – compatibility with RDF, OWL and SWRL
    • also major input to W3C’s upcoming RIF
  – availability of tools
    • OO jDREW, Mandarax, NxBRE, …
  – wide variety of features
    • negation-as-failure, data types, weights, etc.
Step 5: Display results

- **GUI**
  - **ACE**
  - **RuleML, ...**
  - **query (ACE + params)**
  - **DRS, ...**

- **DRS Parser**
  - **RuleML**
  - **JavaCC**

- **DRS grammar**

**TRANSLATOR**
Future Work

- Support new extensions in ACE 5
  - modality
    - If a student procrastinates and an assignment's due date is near then the student must work quickly.
    - If the student misses the due date then he can only beg the professor for an extension.
  - negation as failure and passive voice
    - If a transaction is not recorded by the bank then it is not provable that the transaction happens.

- Investigate adding option for “non-flat” notation
- Extend TRANSLATOR to be bidirectional (also capable of “verbalizing” rules)
Conclusion

• TRANSLATOR allows non-experts to write facts and rules for the Semantic Web
  – critical factor in success of original Web?

• Automated mapping from controlled English input to formal representation
  – ACE \rightarrow DRS \rightarrow RuleML

• Ongoing development by Attempto team
For more information

dhirtle@cs.uwaterloo.ca

http://www.ruleml.org/translator
(includes Java Web Start demo)

http://www.ifi.unizh.ch/attempto/tools