TRANSLATOR:
A TRANSlator from LAnguage TO Rules

David Hirtle

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

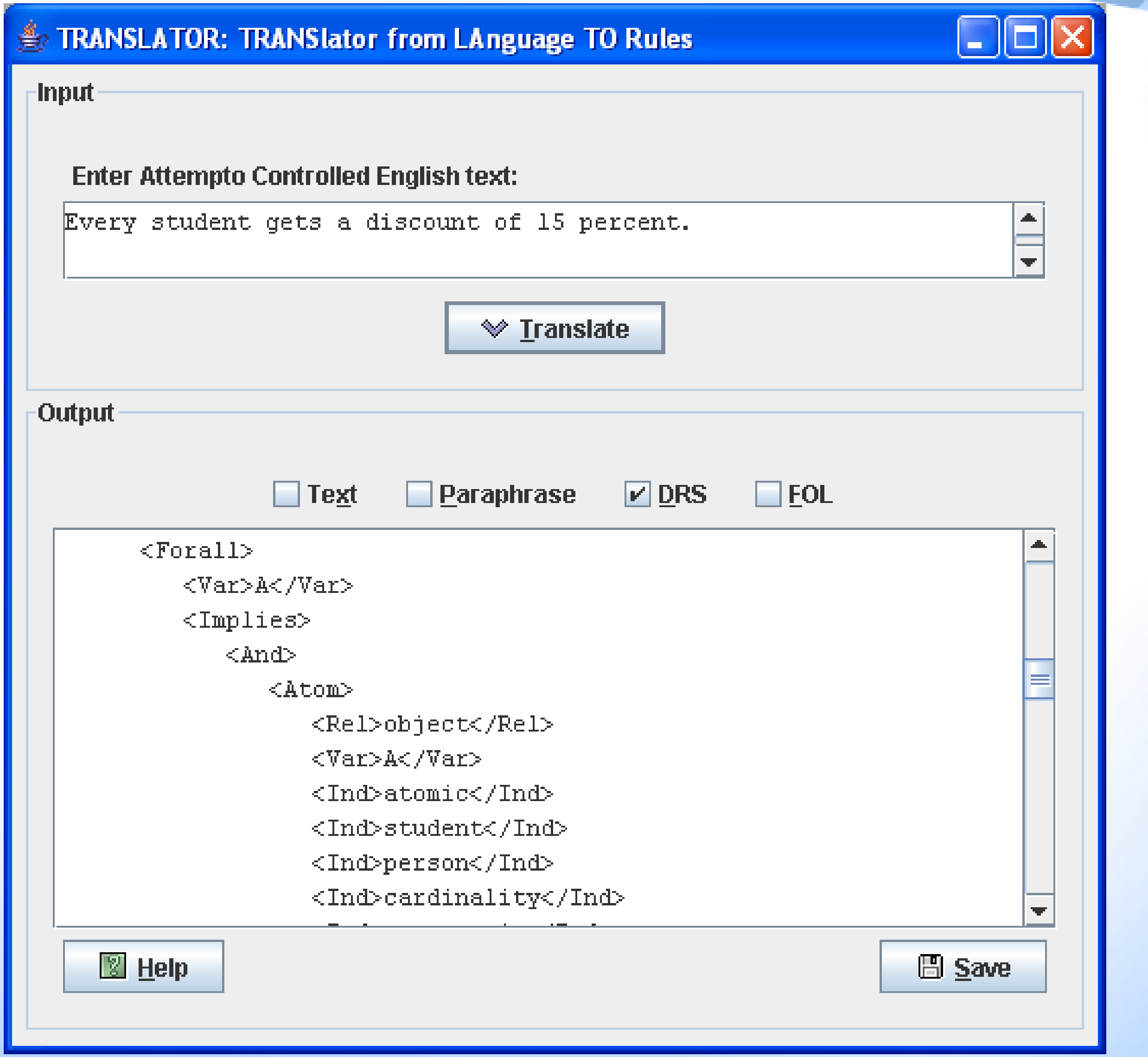
Every student gets a discount of 15 percent.

```
<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

TRANSLATOR



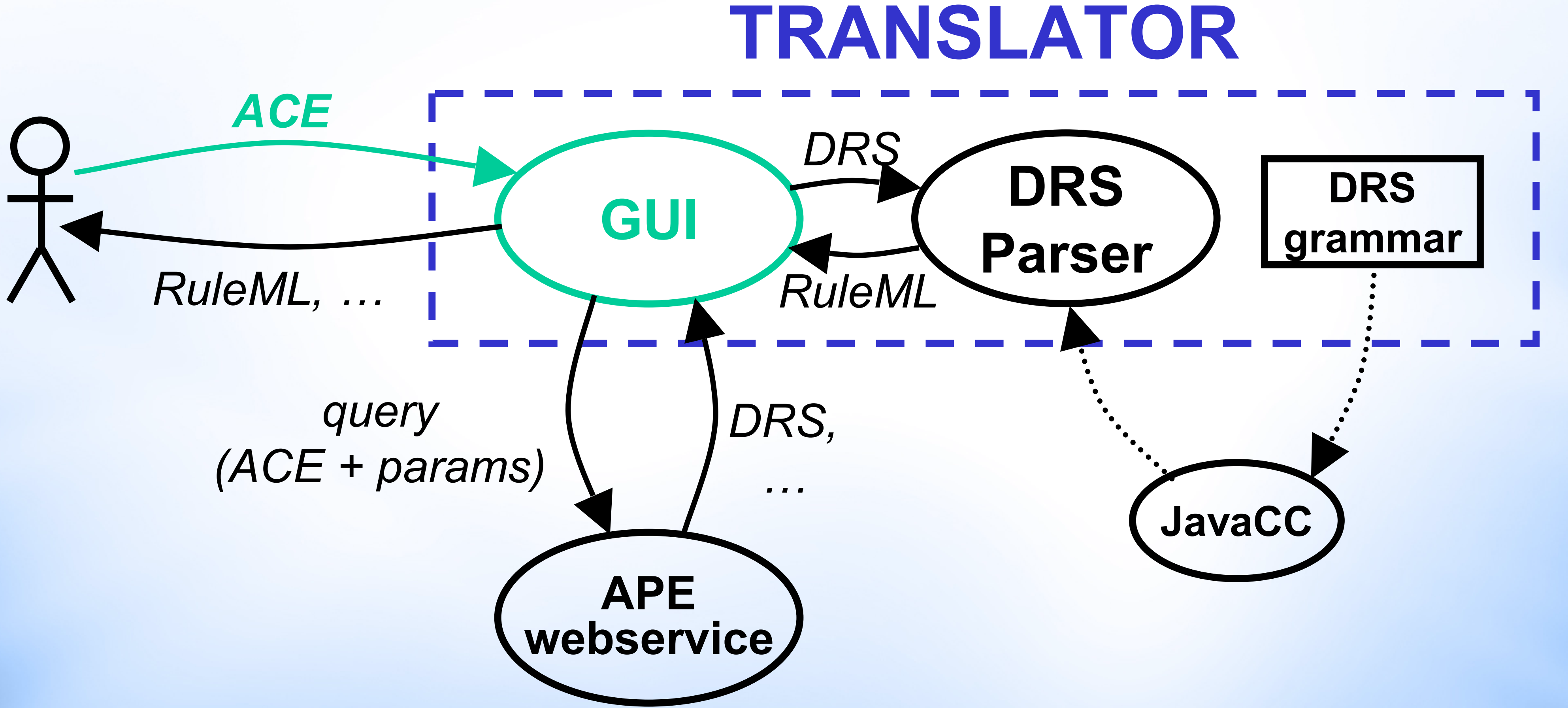
TRANSLATOR: TRANSlator from LAnguage TO Rules
Input
Enter Attempto Controlled English text:
Translate
Output
□ Text
□ Paraphrase
☑ DRS
□ FOL
Help
Save

```
Every student gets a discount of 15 percent.
```

```
    <Forall>
       <Var>A</Var>
       <Implies>
          <And>
             <Atom>
                <Rel>object</Rel>
                <Var>A</Var>
                <Ind>atomic</Ind>
                <Ind>student</Ind>
                <Ind>person</Ind>
                <Ind>cardinality</Ind>
```

Java Web Start

Step 1: Input


TRANSLATOR
ACE
RuleML, …
GUI
DRS
RuleML
DRS
Parser
DRS
grammar
query
(ACE + params)
DRS,
…
APE
webservice
JavaCC

Attempto Controlled English (1)

- 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

Attempto Controlled English (2)

- 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.*

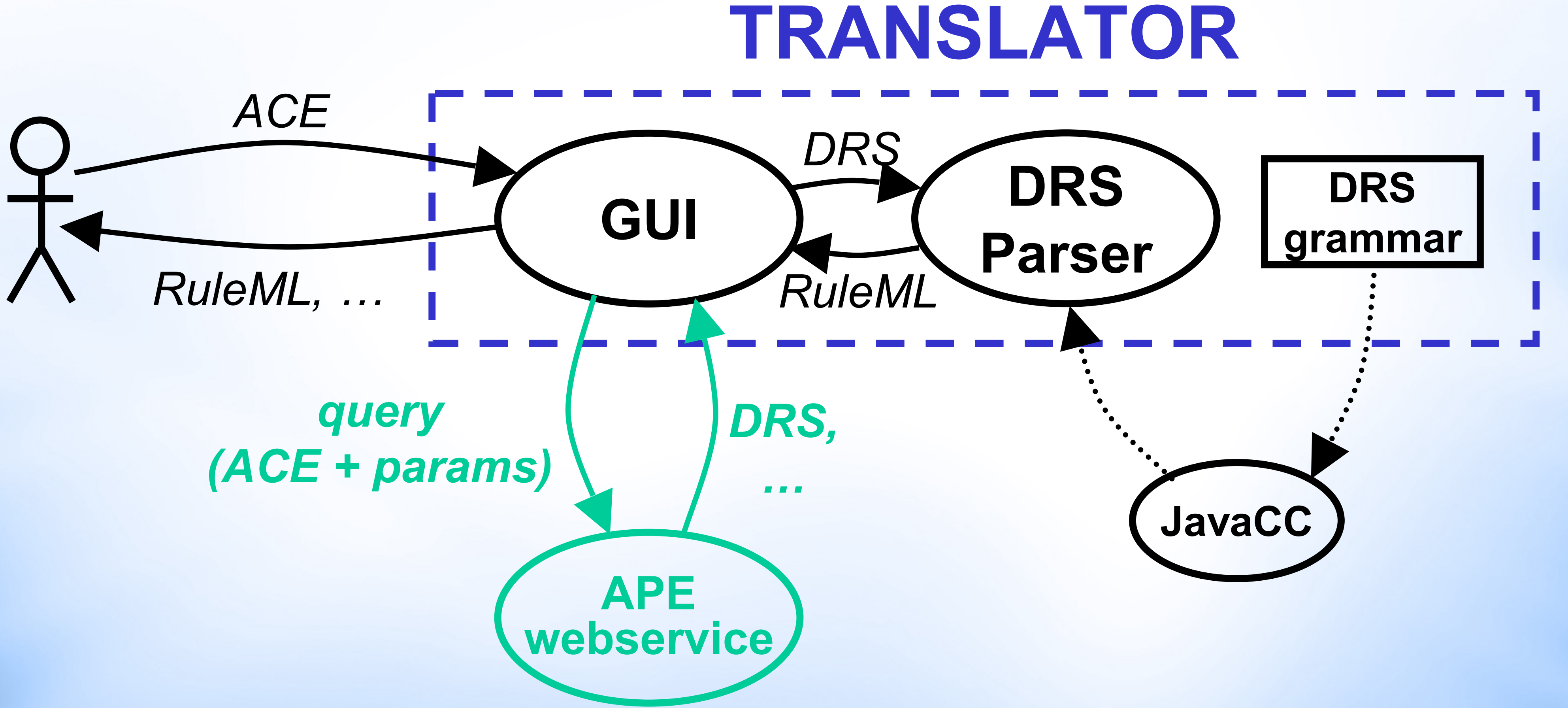
Attempto Controlled English (3)

- 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.

Attempto Controlled English (4)

- 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)` → `object(X,…,student,…)`

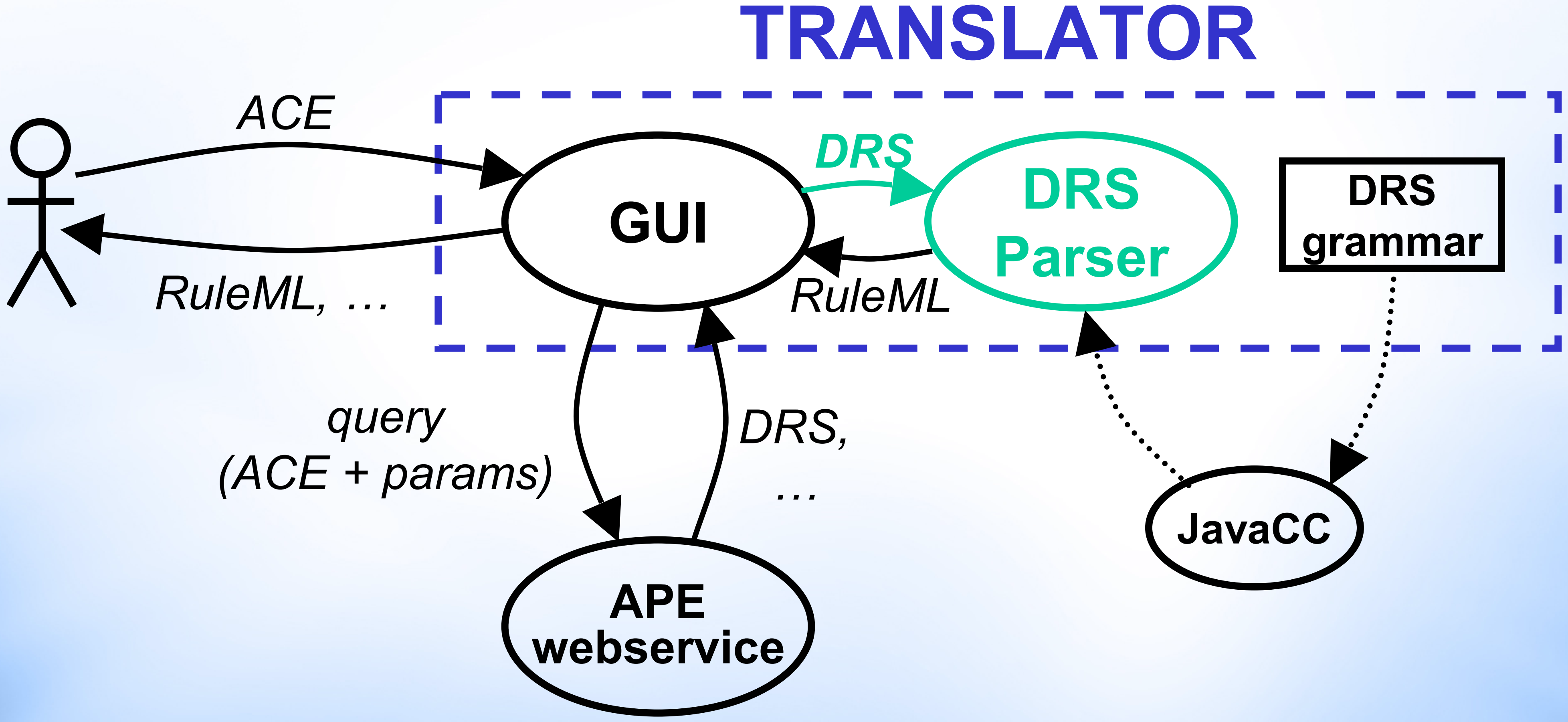
(ACE)

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

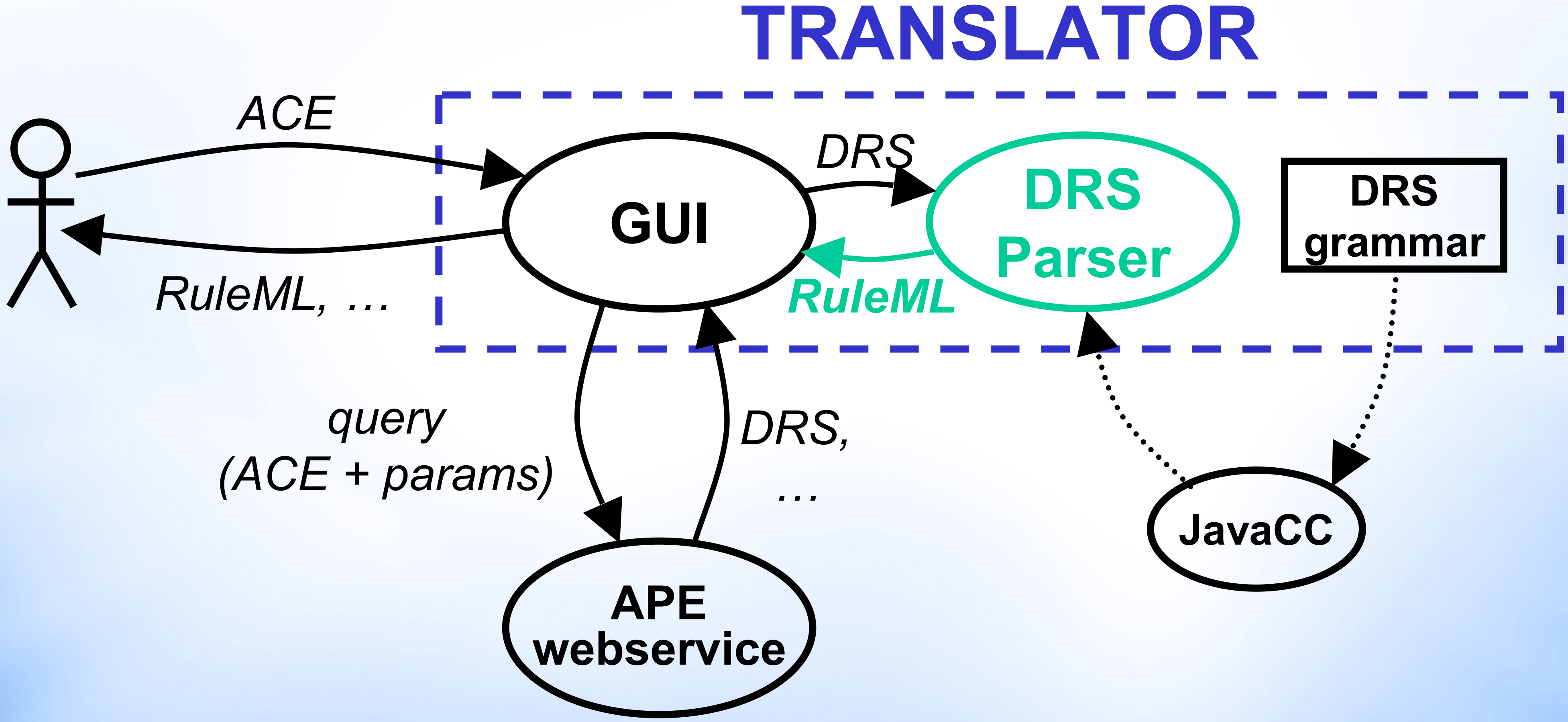
```
[]
    [A]
    object(A, atomic, student, person, cardinality, count_unit, eq, 1)-1
    property(A, honest)-1
        NOT
        [B]
        predicate(B, unspecified, procrastinate, A)-1
    =>
    [C, D, E, F]
    object(C, atomic, mark, object, cardinality, count_unit, eq, 1)-1
    property(C, good)-1
    predicate(D, unspecified, receive, A, C)-1
    predicate(E, unspecified, pass, A, F)-1
    modifier(E, manner, none, easily)-1
    object(F, atomic, course, object, cardinality, count_unit, eq, 1)-1
```

(DRS)

Step 3: Parse DRS



Step 4: Map to RuleML



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., quantifers: `<Forall>`, `<Exists>`
- Reversible
  - – enables future rules → English extension

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

```
[]
   [A]

   object(A, … student, …)
   property(A, honest)
      NOT

      [B]
      predicate(B, … procrastinate, A)


   =>

   [C, D, E, F]

   object(C, … mark, …)
   property(C, good)
   predicate(D, … receive, A, C)
   predicate(E, … pass, A, F)
   modifier(E, manner, … easily)
   object(F, … course, …)
```

(DRS)

```
…
<Forall>
 <Var>A</Var>
 <Implies>
  <And>
   <Atom><Rel>object</Rel>…<Ind>student</Ind>…</Atom>
   <Atom><Rel>property</Rel>…<Ind>honest</Ind></Atom>
   <Neg>
    <Exists>
     <Var>B</Var>
     <Atom><Rel>predicate</Rel>…<Ind>procrastinate</Ind> …
    </Exists>
   </Neg>
  </And>
  <Exists>
   <Var>C</Var><Var>D</Var><Var>E</Var><Var>F</Var>
   <And>
    <Atom><Rel>object</Rel>…<Ind>mark</Ind>…</Atom>
    <Atom><Rel>property</Rel>…<Ind>good</Ind></Atom>
    <Atom><Rel>predicate</Rel>…<Ind>receive</Ind>…</Atom>
    <Atom><Rel>predicate</Rel>…<Ind>pass</Ind>…</Atom>
    <Atom><Rel>modifier</Rel>…<Ind>easily</Ind></Atom>
    <Atom><Rel>object</Rel>…<Ind>course</Ind>…</Atom>
   </And>
  </Exists>
 </Implies>
</Forall>
…
```

(RuleML)

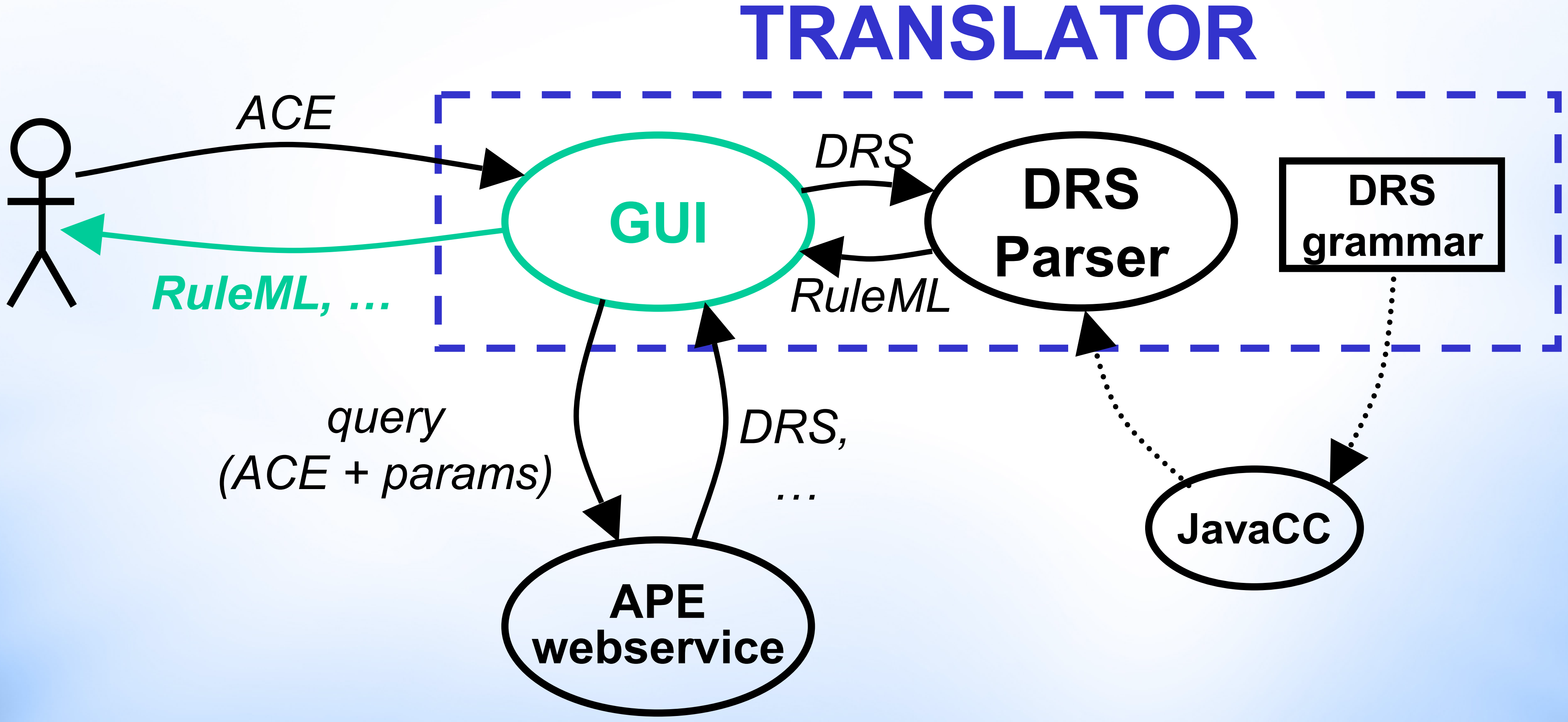
Rule Markup Language (1)

- 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



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 → DRS → 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