Attempto Controlled English: Language, Tools and Applications

Getting Started

Norbert E. Fuchs, Kaarel Kaljurand

Department of Informatics & Institute of Computational Linguistics
University of Zurich

{fuchs, kalju}@ifi.unizh.ch
http://www.ifi.unizh.ch/attempto

December 2006

Overview

- Languages for Knowledge Representation
- Attempto Controlled English (ACE)
- ACE in a Nutshell
- Translating ACE into First-Order Logic
- Typical Applications of ACE

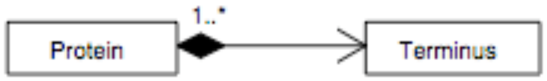
The Problem

- represent as concisely as possible the fact that every protein has a terminus

- questions to be answered
  - who is the author of the representation?
  - for which audience?
  - using which representation, which language?
  - informal or formal representation?
  - representation processable by computer?

Some Solutions

- problem: represent as concisely as possible the fact that every protein has a terminus
- solutions: example representations

first-order logic	$\forall X(protein(X) \rightarrow \exists Y(terminus(Y) \wedge has(X,Y)))$
DL	$Protein \sqsubseteq \exists has.Terminus$
OWL (RDF/XML)	`<owl:Class rdf:ID="Protein">` `<rdfs:subClassOf>` `<owl:Restriction>` `<owl:onProperty rdf:resource="#has"/>` `<owl:someValuesFrom rdf:resource="#Terminus"/>` `</owl:Restriction>` `</rdfs:subClassOf>` `</owl:Class>`
UML	
ACE	Every protein has a terminus.

Languages for Knowledge Representation

- formal languages
  - + well defined-syntax, unambiguous semantics
  - + support automated reasoning
  - – conceptual distance to application domain
  - – incomprehensibility, acceptance problems
- natural language
  - + user-friendly: easy to use and understand
  - + no extra learning effort
  - + high expressiveness, close to application domain
  - – ambiguity, vagueness, incompleteness, inconsistency

Attempto Controlled English (ACE)

- Attempto Controlled English combines pros of formal and natural languages
- ACE is a *controlled* natural language
  - precisely defined, tractable subset of full English
  - automatic, unambiguous translation into first-order logic
- ACE is human *and* machine understandable
  - ACE seems completely natural, but is a formal language
  - ACE is a first-order logic language with an English syntax
- ACE *combines* natural language with formal methods
  - easier to learn and to use than visibly formal languages
  - automated reasoning with ACE via existing tools

An ACE Appetiser
(Actually Quite a Mouthful of ACE)

...

Every customer has at least two cards and their associated codes. If a customer C approaches an automatic teller and she inserts her own card that is valid carefully into the slot and types the correct code of the card then the automatic teller accepts the card and displays a message "Card accepted" and C is satisfied. No card that does not have a correct code is accepted. It is not the case that a customer's card is valid, and is expired or is cancelled. If there is someone X and it is not provable that X is a criminal then the bank can safely assume that X is trustworthy.

...

Important Notice

- The Attempto system does not contain any a priori knowledge of application domains or of formal methods. Users must explicitly define all domain knowledge – for instance definitions, constraints, ontologies – as ACE texts.
- Words occurring in ACE texts are processed by the Attempto system as uninterpreted syntactic elements, i.e. any interpretation of these words is solely performed by the human writer or reader.

The Language ACE

- vocabulary
- grammar
  - construction rules
  - interpretation rules
- style guide

Vocabulary

- predefined function words (articles, prepositions, …)
- predefined phrases ('there is a ...', 'it is false that ...')
- user-defined content-words (nouns, verbs, …)
- basic lexicon (100'000 words)
- optionally: user-defined lexicons
- unknown words: guessing of word class
- unknown words: prefixing with word class

  n:kitkat, *p:Thomas*, *v:google*, *a:trusted*, *a:undeviatingly*

Grammar

- construction rules
  - define admissible sentence structures
  - avoid ambiguous or imprecise constructions

- interpretation rules
  - control logical analysis of admissible sentences
  - resolve remaining ambiguities

Construction Rules: ACE Texts & Queries

- ACE texts
  - sequence of anaphorically interrelated sentences
  - simple sentences
  - composite sentences built recursively from simpler sentences by
    - coordination
    - subordination
    - quantification
    - negation

- interrogative sentences
  - yes/no queries
  - wh-queries
  - sequence of declarative sentences followed by one interrogative sentence

Construction Rules: Simple Sentences

- describe that something is the case – a fact, an event, a state
- structure: subject + verb + (complements) + {adjuncts}
- examples
  A customer waits.
  The temperature is -2.
  A customer inserts 2 cards into a slot.
  A card and a code are valid.

Construction Rules: Elaborating Simple Sentences

- adding adjectives
  A patient customer waits.

- adding possessive nouns and of-prepositional phrases
  John's customer inserts a card of Mary.

- adding strings and variables in apposition
  A machine X print the message "error".

- adding relative phrases
  A customer who is impatient inserts a card that is invalid.

- adding adverbs and prepositional phrases
  A customer manually inserts a card into a slot.

Construction Rules: Composite Sentences

- coordination by *and* is possible between sentences and phrases of the same syntactic type

  A customer waits and a clerk works.

  A customer enters a card and types a code.

  An old and trusted customer enters a card and a code.

- coordination by *or* is possible between sentences, verb phrases and relative clauses

  A customer waits or he sleeps.

  A customer waits or sleeps.

  A customer inserts a card that is valid or that is damaged.

- coordination by *and* and *or* is governed by the standard binding order; commas can be inserted to override the standard binding order

  A customer inserts a Visacard or inserts a Mastercard, and types a code.

Construction Rules: Composite Sentences

- subordination: relative phrases
  A customer who is tired waits.

- subordination: if-then sentences
  If a customer waits then he sleeps.

- subordination: modality
  If a customer can insert a card then he must type a code.
  It is possible/necessary that a customer enters a card.

- *sentence subordination*
  A customer believes that his own card is valid.
  It is true/false that a card is valid.

Construction Rules: Composite Sentences

- existential and universal quantification
  A customer waits
  Every customer waits.
  Every customer inserts a card.
  A customer inserts every card.

- negation
  A customer does not wait.
  A card is not valid.
  No customer waits.
  It is false that a customer waits.

Interpretation Rules: Constructive Disambiguation

- construction rules avoid many ambiguities
- interpretation rules ⇒ deterministic interpretation
- paraphrase reflects interpretation to user
- rephrase input to get alternative interpretations

Input 1:	*A customer inserts a card that is valid and has a code.*
Paraphrase 1:	*A card B is valid. A customer A inserts the card B. The customer A has a code C.*
Input 2:	*A customer inserts a card that is valid and that has a code.*
Paraphrase 2:	*A card B is valid. A customer A inserts the card B. The card B has a code C.*

Interpretation Rules: Constructive Disambiguation

- propositional phrases modify the verb not the noun
  A man {sees a girl with a telescope}.

- relative clauses modify the immediately preceding noun
  A man sees {a girl that has a telescope}

- textual position of a quantifier open its scope that extends to the end of the (coordinated) sentence
  {Everybody loves {somebody}}. $\forall\exists$
  {There is somebody who loves {everybody}}. $\exists\forall$

Interpretation Rules: Resolution of Anaphors

- anaphors refer to noun phrases only
- anaphors can be definite noun phrases, personal and possessive pronouns and variables
- resolution of anaphors: most recent, most specific, accessible noun phrase that agrees in number and gender

  John has a customer. John inserts his card and types a code X. Bill sees X. He inserts his own card and types the code.

Evaluation of Disambiguation

- advantages of constructive disambiguation
  - automatic and efficient disambiguation
  - no use of contextual knowledge, domain knowledge, ontologies
  - simple, systematic, general, easy to learn interpretation rules
  - reliable, reproducible and thus intelligible behaviour
- open problems
  - rules do not always lead to natural interpretation
  - sometimes result in stilted English
  - Can we control all ambiguities with this strategy?
  - Does strategy scale up to larger fragment of ACE?

Revisiting the ACE Appetiser

...

If a customer C approaches an automatic teller and she inserts her own card that is valid carefully into the slot and types the correct code of the card then the automatic teller accepts the card and displays a message "Card accepted" and C is satisfied.

...

Very Brief Style Guide

- While the ACE parser will unravel any syntactically correct sentence, however complex, *you* may have problems to do so.

- Remember that your text is the only source of information; there is no hidden knowledge.

From ACE to First-Order Logic

- ACE is based on Discourse Representation Theory

  (Kamp & Reyle, From Discourse to Logic, Kluwer Academic Publishers, 1993)

- DRT is a linguistic theory whose central concerns are
  - to assign meaning to natural language texts and discourses
  - to account for the context dependence of meaning

From ACE to First-Order Logic

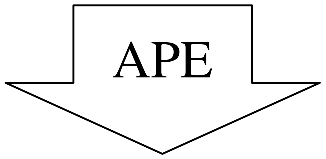
- input: ACE text

  Every company that buys at least 2 standard machines gets a discount.
- target: Extended Discourse Representation Structure (DRS)
  - uses flat syntactic variant of language of standard first-order logic
  - eases encoding of textual relations, e.g. anaphora
  - allows to represent plurals in first-order logic
  - internal representation: `drs(Referents,Conditions)`
- Attempto Parsing Engine (APE)
  - Definite Clause Grammar enhanced with feature structures (Prolog with ProFIT)
  - implements construction and interpretation rules
  - APE generates DRS, syntax tree, paraphrase etc.

Example DRS Representation

ACE Text

Every company that buys at least 2 standard machines gets a discount.



DRS

```
drs([], [drs( [A,B,C],
[object(A,atomic,company,object,cardinality, count_unit,eq,1)-1,
object(B,group,machine,object,cardinality,count_unit,geq,2)-1,
property(B,standard)-1,
predicate(C,event,buy,A,B)-1])
⇒
drs([D, E],
[object(D,atomic,discount,object,cardinality,count_unit,eq,1)-1,
predicate(E,event,get,A,D)-1])])
```

Pretty Printed Example DRS

Every company that buys at least 2 standard machines gets a discount.

```
[]
    [A, B, C]
    object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
    object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
    property(B, standard)-1
    predicate(C, event, buy, A, B)-1
    =>
    [D, E]
    object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
    predicate(E, event, get, A, D)-1
```

Properties of Flat First-Order DRS

Only Predefined Relation Symbols

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Flat First-Order DRS

"Predicates" as Arguments

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Flat First-Order DRS

Lattice-Theoretic Typing of Objects

Every company that buys at least 2 standard machines gets a discount.

```
[]
    [A, B, C]
    object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
    object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
    property(B, standard)-1
    predicate(C, event, buy, A, B)-1
    =>
    [D, E]
    object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
    predicate(E, event, get, A, D)-1
```

Properties of Flat First-Order DRS

Simple Type System

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Flat First-Order DRS

Quantity Information

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object , cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Flat First-Order DRS

Eventuality Types

Every company that buys at least 2 standard machines gets a discount.

```
[]
    [A, B, C]
    object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
    object(B, atomic, machine, object, cardinality, count_unit, geq,2)-1
    property(B, standard)-1
    predicate(C, event, buy, A, B)-1
    =>
    [D, E]
    object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
    predicate(E, event, get, A, D)-1
```

Properties of Flat First-Order DRS

Indices for Tracking in RACE

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, atomic, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Evaluation of Representation

- advantages of flat first-order DRS representation
  - DRS: integrate discourse anaphora
  - first-order: eases automated deduction and reusability
  - flat: possible quantification over "predicates" in first-order logic
  - plurals: represent plurals in first-order logic
  - optional: translation into other first-order languages, e.g. standard or clausal form of first-order logic
- disadvantages
  - we eliminated those that we encountered
  - others?

Applications of ACE

- *Specifications*: automated teller machine, Kemmerer's library data base, Schubert's Steamroller, data base integrity constraints, Kowalski's subway regulations etc.
- *Natural language interfaces*: model generator EP Tableaux (Munich), FLUX agent/robot control (Dresden), MIT's process query language (Zurich), RuleML (New Brunswick)
- *Planned*: medical reports, hospital regulations (Yale)
- *Semantic web*: business & policy rules, translation into and from web-languages, protein ontology (EU Network of Excellence REWERSE)
- *Cooperation Macquarie*: annotations of web-pages in controlled natural language

Specific Application: Reasoning in ACE

- Attempto Reasoner RACE performs deductions on ACE texts
  - RACE shows that one ACE text is the logical consequence of another one
  - RACE answers ACE queries on the basis of an ACE text
  - RACE proves that an ACE text is (in-) consistent
- RACE provides a proof justification in ACE
- RACE finds all proofs
- RACE uses auxiliary first-order axioms and Prolog predicates to reason about plurals, natural numbers, equality

Current & Future Research

- ACE
  - ACE <-> OWL DL
  - prioritised rules
  - support for simulation/execution, imperative mood
  - mathematical structures, e.g. sets, and operations on them
  - decidable subsets of ACE
- RACE
  - improve question answering using models
  - extensions to support modality
  - hypothetical reasoning (“What happens if …?”)
  - abductive reasoning (“Under which conditions …?”)
  - temporal reasoning

Other Controlled Languages

- http://en.wikipedia.org/wiki/Controlled_natural_language
- research
  - Marchiori (W3C): Pseudo Natural Language (Metalog, PNL)
  - Pease & Murray: CELT
  - Pulman (Oxford): Computer Processable Controlled Language, First Order English
  - Schwitter (Macquarie): Processable English (PENG)
  - Skuce (Ottawa): ClearTalk
  - Sowa: Common Logic Controlled English
  - Sukkarieh (Oxford): Controlled Language for Inference Purposes
  - ...
- industry
  - Boeing: Computer-Processable Language (CPL)
  - ...

To Take Home

- ACE is a first-order logic language with the syntax of a subset of English – thus human *and* machine understandable
- ACE does not introduce a division of labour between people who understand formal languages and those who don't – and thus eliminates a major communication problem
- ACE covers the essential part of the semantic continuum

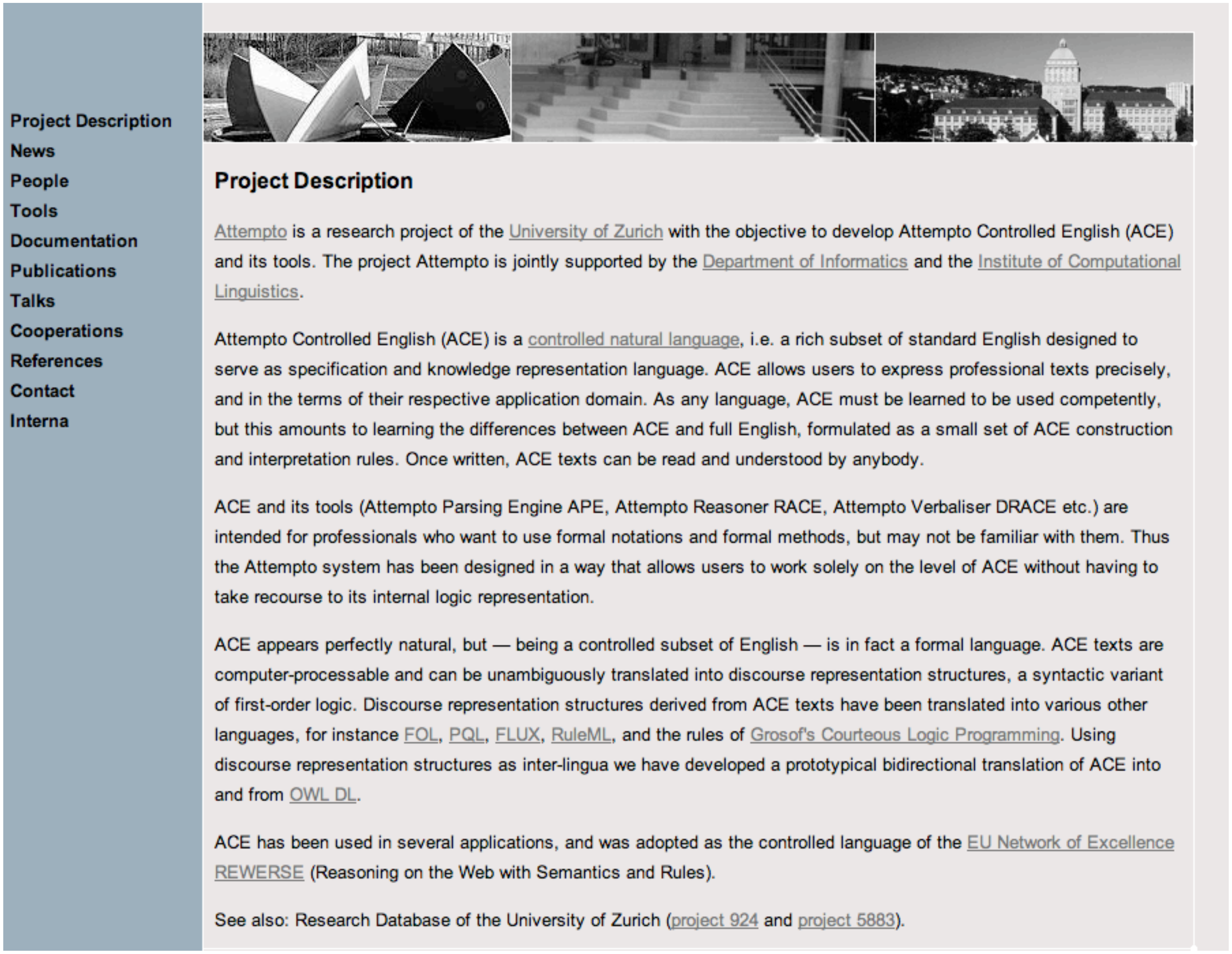
  implicit – informal – formal for humans – formal for machines

  in one and the same notation
- ACE is ontologically neutral, i.e. does not require a priori world knowledge or a domain ontology – though both can be expressed in ACE
- ACE is neutral with regard to particular applications or methods

Attempto Website www.ifi.unizh.ch/attempto

- publications
- documentation
- web-services
- demos
- ... and more

Appendix



Project Description

Attempto is a research project of the University of Zurich with the objective to develop Attempto Controlled English (ACE) and its tools. The project Attempto is jointly supported by the Department of Informatics and the Institute of Computational Linguistics.

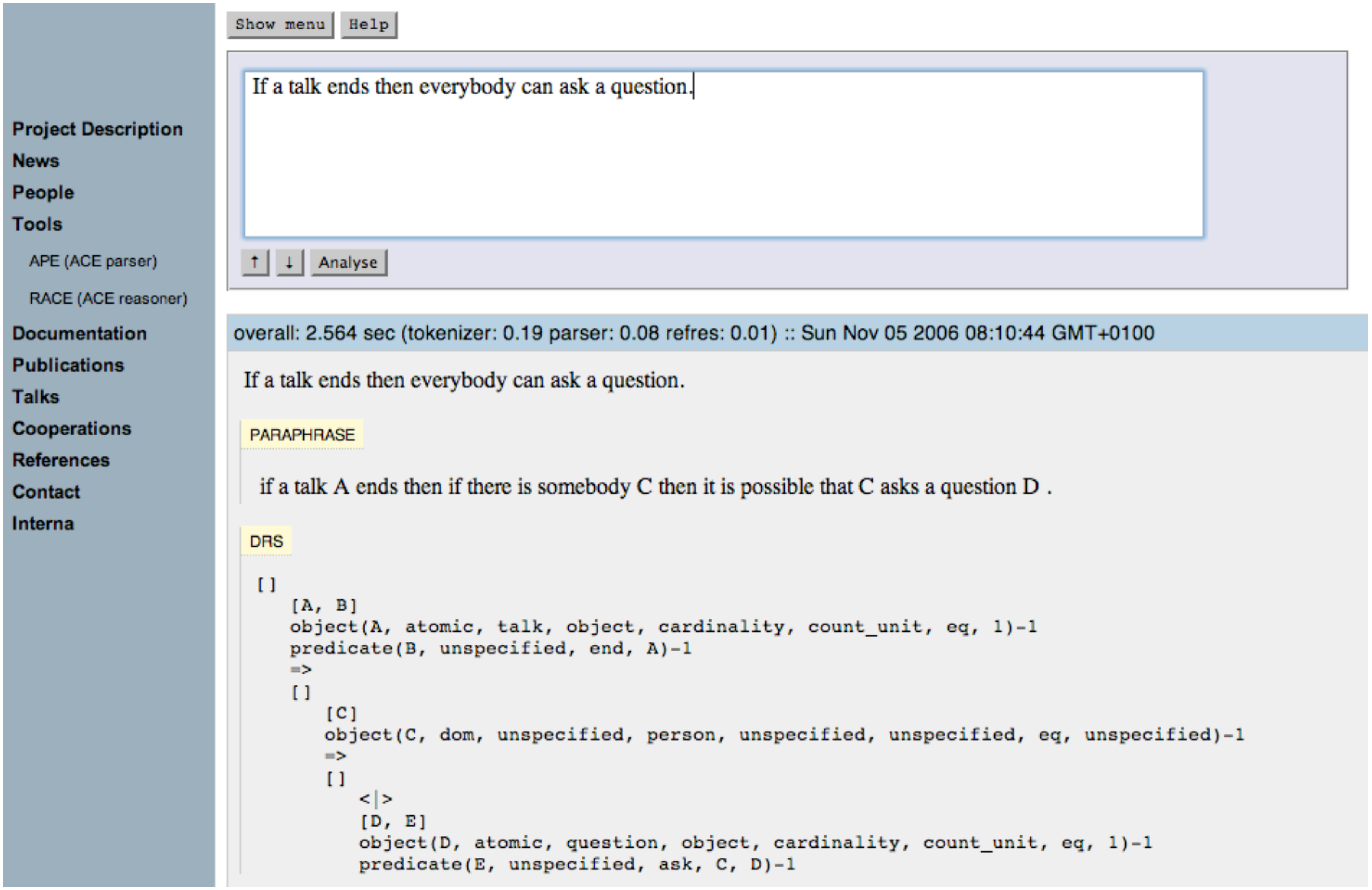
Attempto Controlled English (ACE) is a controlled natural language, i.e. a rich subset of standard English designed to serve as specification and knowledge representation language. ACE allows users to express professional texts precisely, and in the terms of their respective application domain. As any language, ACE must be learned to be used competently, but this amounts to learning the differences between ACE and full English, formulated as a small set of ACE construction and interpretation rules. Once written, ACE texts can be read and understood by anybody.

ACE and its tools (Attempto Parsing Engine APE, Attempto Reasoner RACE, Attempto Verbaliser DRACE etc.) are intended for professionals who want to use formal notations and formal methods, but may not be familiar with them. Thus the Attempto system has been designed in a way that allows users to work solely on the level of ACE without having to take recourse to its internal logic representation.

ACE appears perfectly natural, but — being a controlled subset of English — is in fact a formal language. ACE texts are computer-processable and can be unambiguously translated into discourse representation structures, a syntactic variant of first-order logic. Discourse representation structures derived from ACE texts have been translated into various other languages, for instance FOL, PQL, FLUX, RuleML, and the rules of Grosof's Courteous Logic Programming. Using discourse representation structures as inter-lingua we have developed a prototypical bidirectional translation of ACE into and from OWL DL.

ACE has been used in several applications, and was adopted as the controlled language of the EU Network of Excellence REWERSE (Reasoning on the Web with Semantics and Rules).

See also: Research Database of the University of Zurich (project 924 and project 5883).



If a talk ends then everybody can ask a question.

overall: 2.564 sec (tokenizer: 0.19 parser: 0.08 refres: 0.01) :: Sun Nov 05 2006 08:10:44 GMT+0100

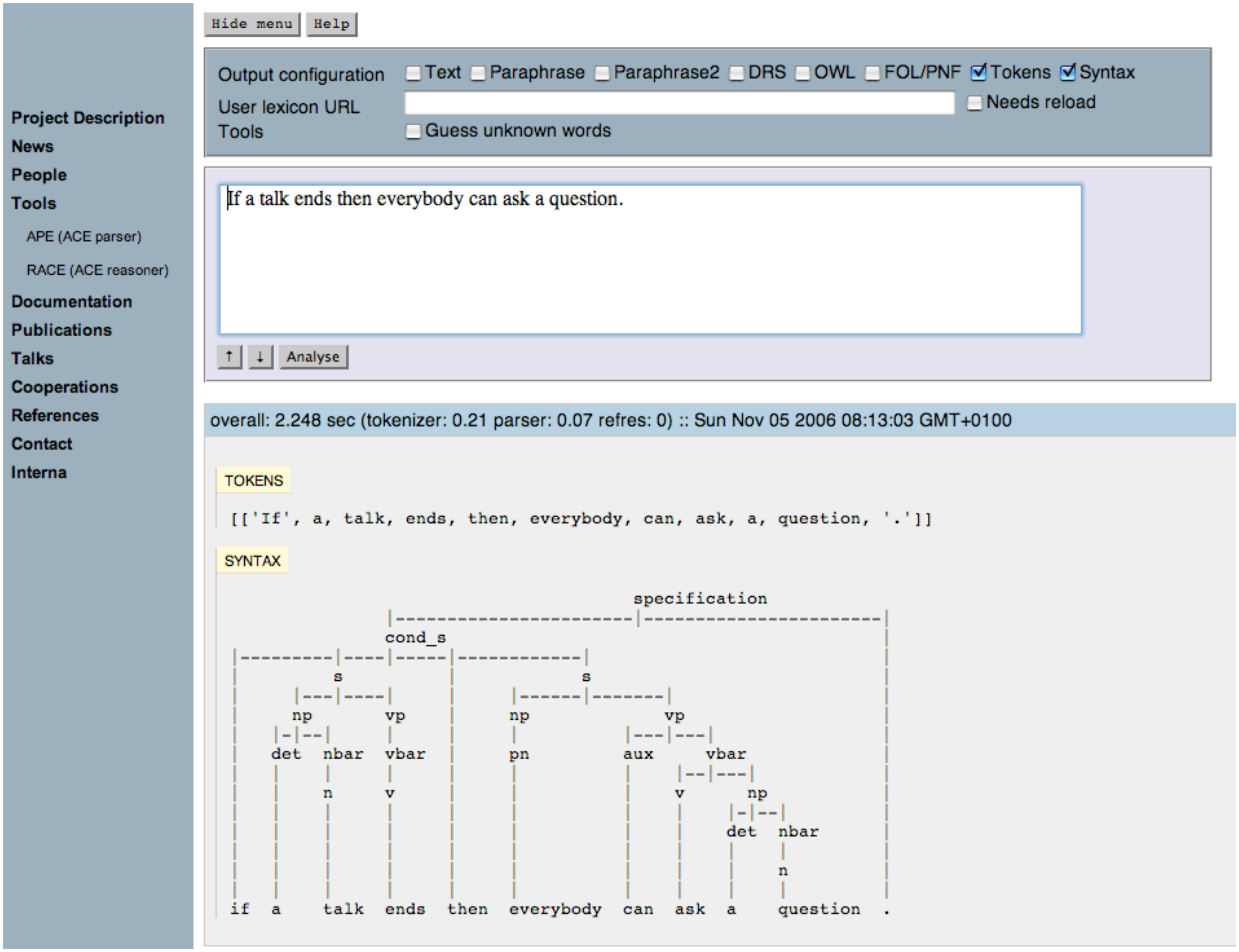
If a talk ends then everybody can ask a question.

PARAPHRASE

if a talk A ends then if there is somebody C then it is possible that C asks a question D .

DRS

```
[]
   [A, B]
   object(A, atomic, talk, object, cardinality, count_unit, eq, 1)-1
   predicate(B, unspecified, end, A)-1
   =>
   []
      [C]
      object(C, dom, unspecified, person, unspecified, unspecified, eq, unspecified)-1
      =>
      []
         <|>
         [D, E]
         object(D, atomic, question, object, cardinality, count_unit, eq, 1)-1
         predicate(E, unspecified, ask, C, D)-1
```



Output configuration [ ] Text [ ] Paraphrase [ ] Paraphrase2 [ ] DRS [ ] OWL [ ] FOL/PNF [✓ Tokens] [✓ Syntax]

User lexicon URL [ ] Needs reload

Tools [ ] Guess unknown words

If a talk ends then everybody can ask a question.

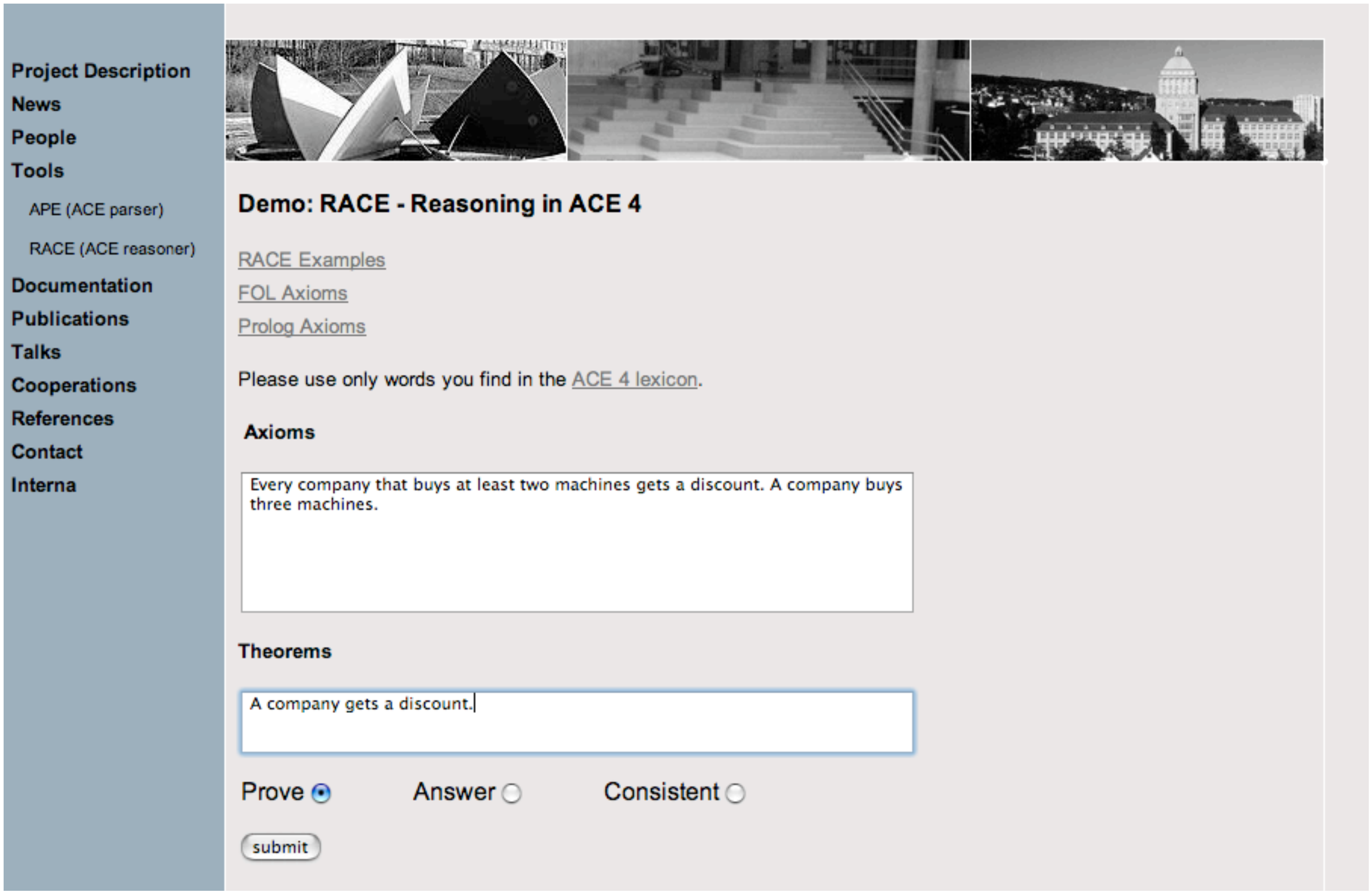
overall: 2.248 sec (tokenizer: 0.21 parser: 0.07 refres: 0) :: Sun Nov 05 2006 08:13:03 GMT+0100

TOKENS

```
[['If', a, talk, ends, then, everybody, can, ask, a, question, '.']]
```

SYNTAX

```
                                       specification
               |-----------------------|-----------------------|
               cond_s                                          |
|---------|----|-----|------------|                            |
|         s          |            s                            |
|     |---|----|     |     |------|-------|                    |
|     np       vp    |     np             vp                   |
|   |-|--|     |     |     |          |---|---|                |
|   det  nbar  vbar  |     pn         aux     vbar             |
|   |    |     |     |     |          |    |--|---|            |
|   |    n     v     |     |          |    v      np           |
|   |    |     |     |     |          |    |    |-|--|         |
|   |    |     |     |     |          |    |    det  nbar      |
|   |    |     |     |     |          |    |    |    |         |
|   |    |     |     |     |          |    |    |    n         |
|   |    |     |     |     |          |    |    |    |         |
if  a    talk  ends  then  everybody  can  ask  a    question  .
```



Demo: RACE - Reasoning in ACE 4

RACE Examples
FOL Axioms
Prolog Axioms

Please use only words you find in the ACE 4 lexicon.

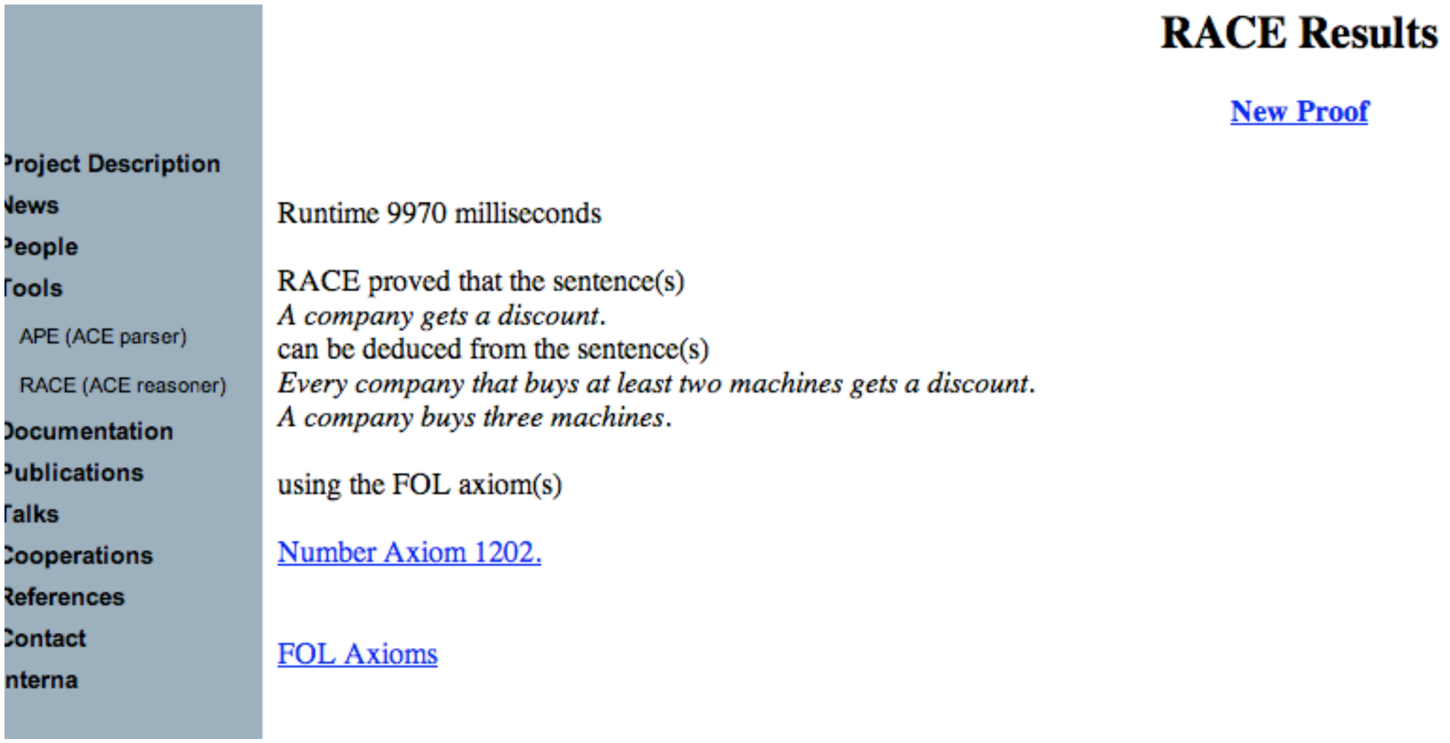
Axioms

Every company that buys at least two machines gets a discount. A company buys three machines.

Theorems

A company gets a discount.

[✓ Prove] Answer Consistent



RACE Results

New Proof

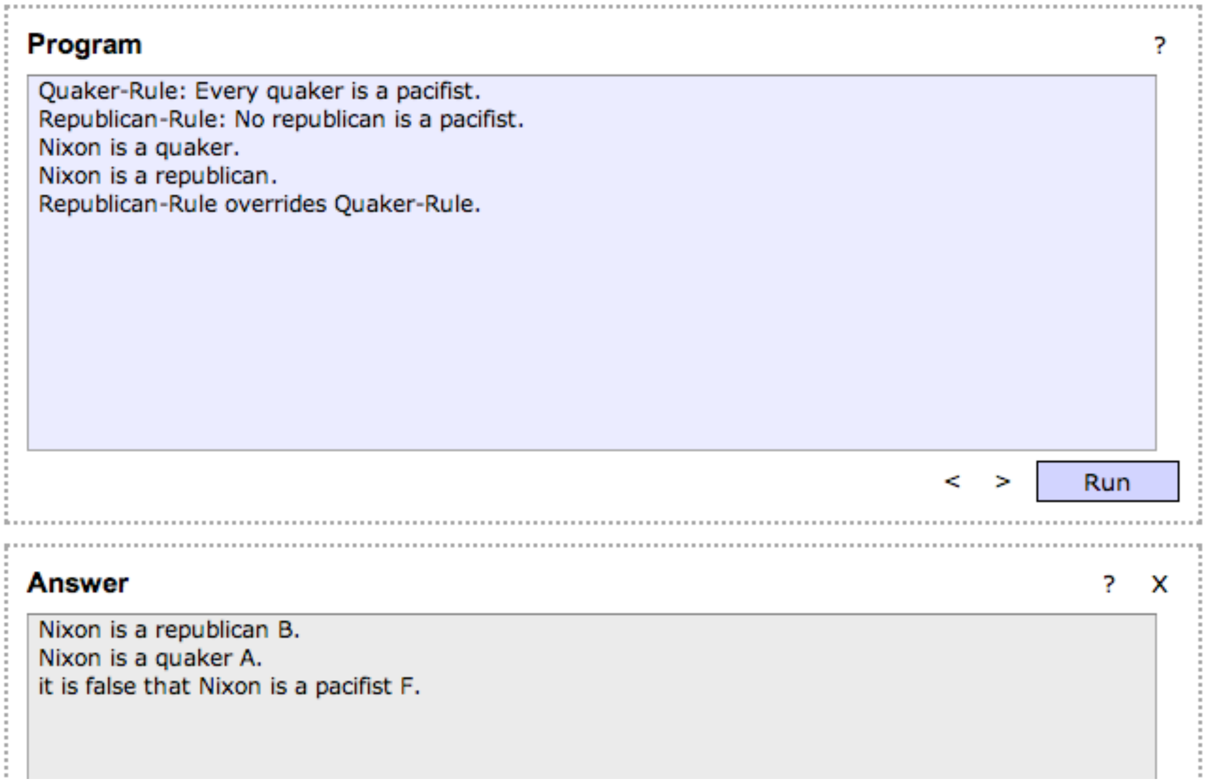
Runtime 9970 milliseconds

RACE proved that the sentence(s)
A company gets a discount.
can be deduced from the sentence(s)
Every company that buys at least two machines gets a discount.
A company buys three machines.

using the FOL axiom(s)

Number Axiom 1202.

FOL Axioms


Program

?

```
Quaker-Rule: Every quaker is a pacifist.
Republican-Rule: No republican is a pacifist.
Nixon is a quaker.
Nixon is a republican.
Republican-Rule overrides Quaker-Rule.
```

Answer

```
Nixon is a republican B.
Nixon is a quaker A.
it is false that Nixon is a pacifist F.
```