Knowledge Representation and Reasoning in (Controlled) Natural Language

Norbert E. Fuchs

Department of Informatics & Institute of Computational Linguistics
University of Zurich

fuchs@ifi.unizh.ch
http://www.ifi.unizh.ch/attempto

University of Washington

December 2005

Lewis Carroll as a Grocer and a Person

Given the sentences

Every grocer is a person.

Lewis Carroll is a grocer.

show that

Lewis Carroll is a person.

Trivial. Just apply modus ponens once.

Lewis Carroll's Grocer Puzzle

Given the sentences

Every honest and industrious person is healthy.

No grocer is healthy.

Every industrious grocer is honest.

Every cyclist is industrious.

Every unhealthy cyclist is dishonest.

No healthy person is unhealthy.

No honest person is dishonest.

Every grocer is a person.

Every cyclist is a person.

show that

No grocer is a cyclist.

That's getting kind of hairy.
Let me think ...

Invoking RACE

Submitting Carroll's Grocer Puzzle to the Automatic Reasoner RACE we get

Runtime 10 milliseconds

RACE proved that the sentence(s)

No grocer is a cyclist.

can be deduced from the sentence(s)

Every honest and industrious person is healthy.

No grocer is healthy.

Every industrious grocer is honest.

Every cyclist is industrious.

Every cyclist is a person.

That's not fair. I am still thinking.
Fuchs must be using some tricks.

Trick # 1
This is Not English

Every honest and industrious person is healthy.
No grocer is healthy.
Every industrious grocer is honest.
Every cyclist is industrious.
Every unhealthy cyclist is dishonest.
No healthy person is unhealthy.
No honest person is dishonest.
Every grocer is a person.
Every cyclist is a person.
No grocer is a cyclist.

Trick # 2
RACE Does Not Reason in English

Overview

- Languages for Knowledge Representation
- Attempto Controlled English (ACE)
- Translating ACE into First-Order Logic
- Automated Reasoning in ACE (RACE)
- Applications
- Conclusions

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

- Can we combine the pros of formal and natural languages?

Attempto Controlled English (ACE)

- 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 it is a formal language
  - ACE is a first-order logic language with an English syntax
  - while the meaning of a sentence in natural language can vary depending on its – possibly only vaguely defined – context, the meaning of an ACE sentence is completely and uniquely defined
- ACE *combines* natural language with formal methods
  - easier to learn and use than visibly formal languages
  - automated reasoning with ACE via existing tools

The Language ACE

- Vocabulary
  - predefined function words (articles, prepositions, …)
  - predefined phrases ('there is a ...', 'it is not the case that ...')
  - user-defined content-words (nouns, verbs, …)
  - basic lexicon (100'000 words), user-defined lexicons
- Construction rules
  - define admissible sentence structures
  - avoid ambiguous or imprecise constructions
- Interpretation rules
  - control logical analysis of admissible sentences
  - resolve remaining ambiguities

Example: Construction Rules

- Simple Sentences

subject	predicate	(complements)	{adjuncts}
A customer	*waits.*		
A customer	*inserts*	*a valid card*	*into a machine.*

- Composite sentences built with predefined constructors

If a customer inserts a card that is valid then
the automatic teller accepts the card and types a message.

- Verbs
  - third person, indicative, simple present, active
  - no modality (*can*, *must*) or intensionality (*believe*)

Example: Interpretation Rules

- Prepositional phrases modify the verb not the noun

  A customer { enters a card with a code} .

- Relative clauses modify the immediately preceding noun

  A customer enters {a card that has a code}.

- Surface position of a quantifier determines its relative scope

  A customer types every code. ∃∀

  Every customer enters a card. ∀∃

- Anaphora: most recent, most specific, accessible noun phrase

  John is a customer. He inserts a card that belongs to himself and types a code. Bill sees it. He inserts his own card and types the code.

Example: Interpretation Rules

- Prepositional phrases modify the verb not the noun

  A customer { enters a card with a code} .

- Relative clauses modify the immediately preceding noun

  A customer enters {a card that has a code}.

- Surface position of a quantifier determines its relative scope

  A customer types every code. ∃∀

  Every customer enters a card. ∀∃

- Anaphora: most recent, most specific, accessible noun phrase

  John is a customer. He inserts a card that belongs to himself and types a code. Bill sees it. He inserts his own card and types the code.

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 customer inserts {a card that is valid} and has a code.*

Input 2: *A customer inserts a card that is valid and <u>that</u> has a code.*

Paraphrase 2: *A customer inserts {a card that is valid and that has a 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?

From ACE to First-Order Logic

- Input: ACE text

  Every company that buys a standard machine gets a discount.
- Target: Extended Discourse Representation Structure (DRS)
  - flat syntactic variant of standard first-order logic
  - eases encoding of textual relations, e.g. anaphora
  - allows to represent plural 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 in Core ACE etc.

Example DRS Representation

ACE Text

Every company that buys a standard machine gets a discount.

DRS

```
drs([],[drs([A,B,C,D,E],[structure(B,atomic)-1,
quantity(B,cardinality,count_unit,A,eq,1)-1, object(B,company)-1,
structure(D,atomic)-1, quantity(D,cardinality,count_unit,C,eq,1)-1,
property(D,standard)-1, object(D,machine)-1,
predicate(E,event,buy,B,D)-1]) => drs([F,G,H],
[structure(G,atomic)-1,quantity(G,cardinality,count_unit,F,eq,1)-1,
object(G,discount)-1, predicate(H,event,get,B,G)-1])])
```

Pretty Printed Example DRS

Every company that buys a standard machine gets a discount.

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

Properties of Flat First-Order DRS

Every company that buys a standard machine gets a discount.

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

only predefined relation symbols

Properties of Flat First-Order DRS

Every company that buys a standard machine gets a discount.

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

only predefined relation symbols

"predicates" as arguments

Properties of Flat First-Order DRS

Every company that buys a standard machine gets a discount.

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

only predefined relation symbols

"predicates" as arguments

eventuality types

Properties of Flat First-Order DRS

Every company that buys a standard machine gets a discount.

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

only predefined relation symbols

“predicates” as arguments

eventuality types

lattice theoretic typing of objects

Properties of Flat First-Order DRS

Every company that buys a standard machine gets a discount.

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

only predefined relation symbols

"predicates" as arguments

eventuality types

lattice theoretic typing of objects

quantity information

Properties of Flat First-Order DRS

Every company that buys a standard machine gets a discount.

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

only predefined relation symbols

“predicates” as arguments

eventuality types

lattice theoretic typing of objects

quantity information

index for tracking within RACE

Evaluation of Representation

- Advantages of flat first-order DRS representation
  - DRS: integrate discourse anaphora
  - first-order: eases automated deduction and reusability
  - flat: 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
- Problems and further research
  - large number of conditions can reduce efficiency
  - increase coverage: set-theory, temporal relations, …

Automated Reasoning with ACE

- Examine ACE text using formal methods, specifically deduction
- Requirements for ACE reasoner RACE
  - input and output in ACE
  - generate *all* proofs
  - give a user-friendly justification of a proof
  - allow for auxiliary first-order axioms
  - interface to evaluable functions and predicates
  - combine theorem proving with model generation
  - hide internal working from casual user

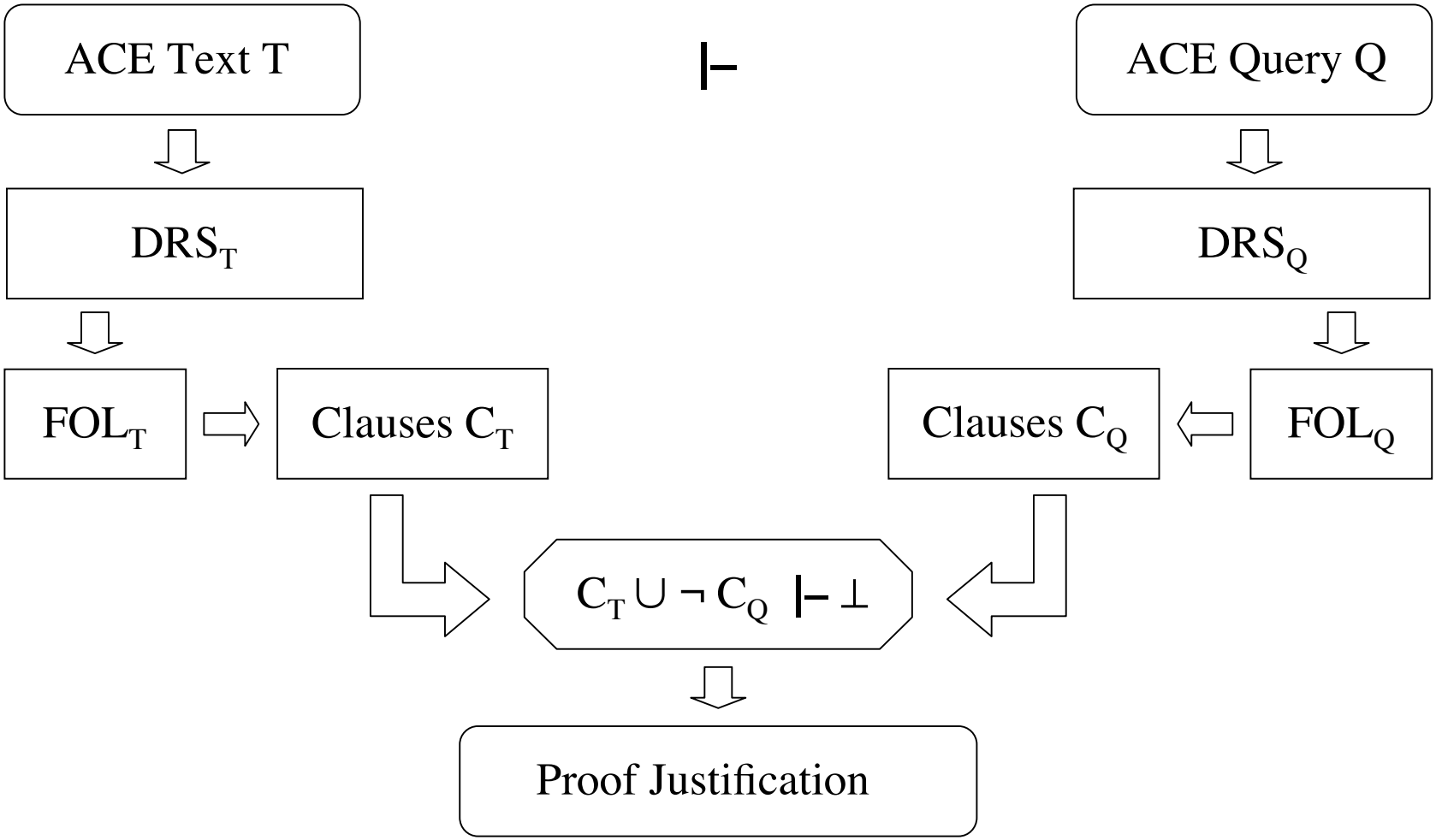
Basis for Attempto Reasoner

- Existing first-order reasoners as potential basis
  - direct DRS deduction, leanTAP, EP Tableaux, Otter, Mace, Satchmo, …
- Satchmo (Manthey & Bry 1988) as basis of RACE
  - combines model generation with theorem proving
  - generates minimal finite models of clauses (if existent)
  - correct for unsatisfiability of range-restricted clauses
  - complete for unsatisfiability if used level-saturated
  - efficient Prolog core
  - allows local extensions and modifications in Prolog

RACE Extensions of Satchmo

- RACE generates all proofs
  - Satchmo stops immediately if it detects unsatisfiability
  - RACE finds *all* minimal unsatisfiable subsets of clauses
- RACE gives a justification of every proof
  - RACE collects indices of logical atoms used for a proof
  - RACE generates for each proof a report showing which ACE sentences were used to derive which ACE query
- Input and output in ACE

Structure of RACE



RACE Detects Entailments

Text *Every company that buys a standard machine gets a discount.*
A British company buys a standard machine.
A French company buys a special machine.

Query *A company gets a discount.*

RACE proved that the sentence(s)
A company gets a discount.
can be deduced from the sentence(s)
Every company that buys a standard machine gets a discount.
A British company buys a standard machine.

RACE Answers Questions

Text *Every company that buys a standard machine gets a discount.*
A British company buys a standard machine.
A French company buys a special machine.

Query *Who buys a machine?*

1. RACE proved that the query (-ies)
   Who buys a machine?
   can be answered on the basis of the sentence(s)
   A British company buys a standard machine.

2. RACE proved that the query (-ies)
   Who buys a machine?
   can be answered on the basis of the sentence(s)
   A French company buys a special machine.

RACE Detects Inconsistencies

Text *Every company that buys a standard machine gets a discount.*
A British company buys a standard machine.
A French company buys a standard machine.
There is no company that gets a discount.

1. RACE proved that the sentence(s)
   Every company that buys a standard machine gets a discount.
   A British company buys a standard machine.
   There is no company that gets a discount.
   are inconsistent.

2. RACE proved that the sentence(s)
   Every company that buys a standard machine gets a discount.
   A French company buys a standard machine.
   There is no company that gets a discount.
   are inconsistent.

Auxiliary Axioms for RACE

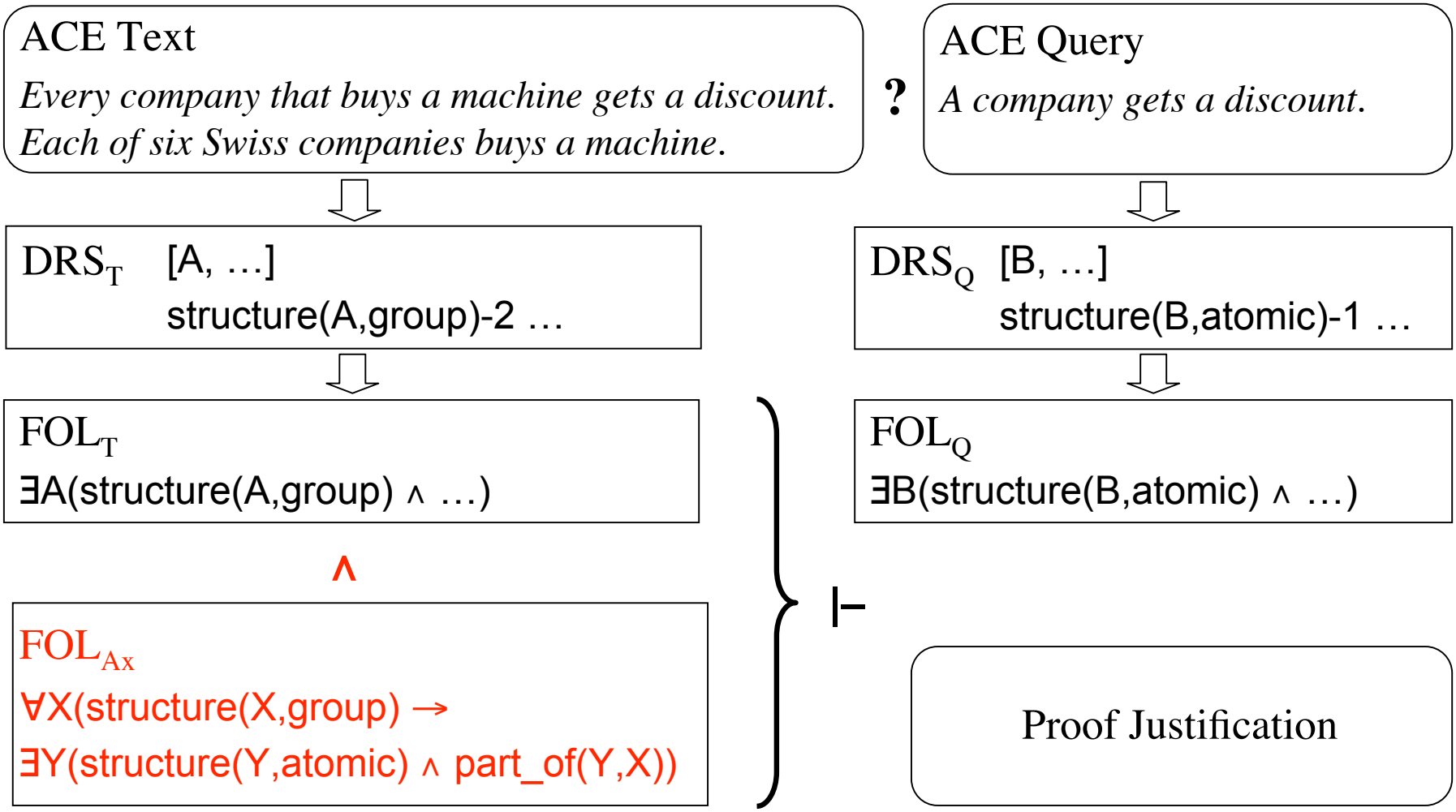
- ACE can express plural sentences

  Six Swiss companies buy a machine. (collective)

  Each of six Swiss companies buys a machine. (distributive)

- First-order representation of plurals
  - additional group objects with lattice-theoretic structure
  - requires additional (domain-independent) knowledge
    - auxiliary axioms for lattice-theory, numbers, equality, …
    - evaluable functions and predicates
- Flat notation allows us to integrate axioms in FOL

Proof with Auxiliary Axiom



Entailment Example with Axioms

Text *Every company that buys a machine gets a discount.*
Each of six Swiss companies buys a machine.

Query *A company gets a discount.*

RACE proved that the sentence(s)
A company gets a discount.
can be deduced from the sentence(s)
Every company that buys a machine gets a discount.
Each of six Swiss companies buys a machine.
using the auxiliary axiom(s)
(Ax. 9): Definition of proper_part_of.
(Ax. 10-1): Every group consists of atomic parts.
(Ax. 22-1): Number Axiom.

Question Answering with Axioms

Text *Each of six Swiss companies buys a machine.*
A German company buys a special machine.

Query *Who buys machines?*

1. RACE proved that the query (-ies) *Who buys machines?* can be answered on the basis of the sentence(s) *Six Swiss companies each buy a machine.* using the FOL axiom(s)	(Ax. 10-2): Groups have atomic parts. (Ax. 2): atomic => dom (Ax. 9): Definition of proper_part_of. (Ax. 11): Atoms have no proper parts. (Ax. 15-1): Identity axiom for objects. (Ax. 22-1): Number Axiom.
2. RACE proved that the query (-ies) *Who buys machines?* can be answered on the basis of the sentence(s) *A German company buys a special machine.* using the FOL axiom(s)	(Ax. 2): atomic => dom (Ax. 11): Atoms have no proper parts. (Ax. 15-1): Identity axiom for objects. (Ax. 22-1): Number Axiom.

Example Auxiliary Axioms

- Internal representation: fol_axiom(Index, Axiom, Text)
- Number of axioms: ca. 80
- (Ax. 10-2): Groups have atomic parts.

  $\forall X(\text{structure}(X,\text{group}) \rightarrow \exists Y(\text{structure}(Y,\text{atomic}) \wedge \text{part\_of}(Y,X)))$
- (Ax. 11): Atoms have no proper parts.

  $\forall X \forall Y(\text{structure}(X,\text{atomic}) \wedge \text{part\_of}(X,Y) \rightarrow \text{is\_equal}(X,Y))$
- (Ax. 15-1): Identity axiom for objects

  $\forall X \forall Y \forall P(\text{object}(X,P) \wedge \text{is\_equal}(X,Y) \rightarrow \text{object}(Y,P))$
- Problems: Satchmo core offers no equality reasoning, some axioms cause inefficiency

Add Evaluable Prolog Predicates

- Problems involving natural numbers …

  Text *Every company that buys at least three machines gets a discount. A German company buys four machines.*

  Query *A company gets a discount.*

- … require knowledge about natural numbers

  ∀X∀C(quantity(X,cardinality,C,eq,4) ⇒ quantity(X,cardinality,C,geq,3))

- Prolog predicate avoids instantiation problems

```
quantity(_A, _Dim, _Unit, Card, geq, NewN):-
        number(NewN),
        quantity(_A, _Dim, _Unit, Card, eq, GivenN),
        number(GivenN),
        NewN =< GivenN.
```

Evaluation of RACE

- User-friendly interface for logic-oriented tasks
- Example Performance: “Steamroller”

Representation	**Standard**	**Attempto DRS**
Satchmo (original)	15 ms	1990 ms
RACE	70 ms	375 ms

- Open problems
  - extended notation decreases performance of reasoner
  - finding *all* solutions increases search space
  - scalability, robustness

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)
- *Partially investigated or suggested applications*: knowledge assimilation (Munich), medical reports (Uppsala), planning (Uppsala/Stockholm), synthesiser of constraint logic programs (Uppsala), ontologies, legal texts, standards, teaching logic
- *Semantic web*: EU Network of Excellence REWERSE (busines and policy rules, protein ontology etc.)

Natural Language for the Semantic Web?

[…] a truly semantic web is more likely to be based on natural language processing than on annotations in any artificial language.

If I had to process web annotations in any artificial language, I would prefer to use controlled English rather than special notations such as RDF. There is no reason why web annotations have to be humanly unreadable in order to be easy to process by a computer.

(John F. Sowa, CG Mailing List, October 19, 2003)

Further Research

- ACE
  - translate ACE into web-languages, e.g. UML, OWL
  - verbalise web-languages, e.g. UML, OWL, in ACE
  - extensions of ACE to support (limited forms of) modality
  - extensions of ACE to support simulation/execution: prioritised rules, negation as failure, variables, relations/functions, imperative
  - Courteous Logic Programming (Grosof et al.)
  - decidable subsets of ACE
- RACE
  - improve question answering using Satchmo models
  - hypothetical reasoning (“What happens if …?”)
  - abductive reasoning (“Under which conditions …?”)
  - temporal reasoning

Related Work

- Website (www.ics.mq.edu.au/~rolfs/controlled-natural-languages)
- Research
  - Marchiori (MIT, Venezia): Pseudo Natural Language
  - Pease & Murray: CELT
  - Pulman (Oxford): Computer Processable Controlled Language, First Order English
  - Schwitter (Macquarie): Processable English
  - Skuce (Ottawa): ClearTalk
  - Sowa: Common Logic Controlled English
  - Sukkarieh (Oxford): Controlled Language for Inference Purposes
  - Esprit Framework IV: Prosper
  - ...
- Industry
  - AECMA
  - Boeing
  - Caterpillar
  - ...

Acknowledgements

- Collaborators of the Project Attempto
  - *Research Staff*: Gerold Schneider, Kaarel Kaljurand
  - *Student Staff*: Tobias Kuhn
  - *Former Collaborators*: Alexandra Bünzli, Marc Dörflinger, Wei Fang, Bernhard Hamberger, Robert Hein, Stefan Höfler, Sabine Koch, Fabio Rinaldi, Uta Schwertel, Rolf Schwitter, Georgios Tagalakis, Sunna Torge, Wamberto Vasconcelos
- Funding of the Projects Attempto and REWERSE
  - University of Zurich
  - Swiss National Science Foundation
  - Swedish National Defence College
  - Swiss State Secretariat for Education and Research

Conclusions

- ACE is a first-order logic language with an English syntax, thus human *and* machine understandable
- ACE covers the essential part of the semantic continuum

  implicit – informal – formal for humans – formal for machines

  in one and the same notation
- RACE is a theorem prover and model generator mainly for ACE, but also accepts FOL and Prolog
- ACE and RACE are general tools
  - not requiring a priori world knowledge or a domain ontology (though both can be expressed in ACE)
  - neutral with regard to particular applications or methods

OK, OK …
But this calls for many questions.

Attempto Website

www.ifi.unizh.ch/attempto/