Attempto Controlled English

Norbert E. Fuchs
Department of Informatics & Institute of Computational Linguistics
University of Zurich, Switzerland
www.ifi.unizh.ch/attempto

Stanford University
December 2005

Overview

- Languages for Knowledge Representation
- Attempto Controlled English (ACE)
- An ACE Appetiser
- Translating ACE into First-Order Logic
- ACE Vocabulary
- ACE Construction Rules
- ACE Interpretation Rules
- Very Brief Style Guide
- Applications of ACE

Languages for Knowledge Representation

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

An ACE Appetiser
(Actually Quite a Mouthful of ACE)

...

Every customer has at least 2 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 happy. 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.

...

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.

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
- target: Extended Discourse Representation Structure (DRS)
  - DRS eases encoding of textual relations, e.g. anaphora
  - extended DRS allows to represent plural sentences
  - extended DRS uses flat syntactic variant of language of first-order logic
  - internal representation: drs(Referents,Conditions)
- Attempto Parsing Engine (APE)
  - Definite Clause Grammar enhanced with feature structures (ProFIT)
  - APE generates
    - syntax tree
    - DRS (plus translation of the DRS into FOL)
    - paraphrase of input in Core ACE (subset of ACE)
    - error messages
    - ...

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, …

The Language ACE

- vocabulary
  - predefined *function words* (articles, prepositions, …)
  - predefined *fixed phrases* ('there is a ...', 'it is not the case that ...')
  - user-defined *content words* (nouns, verbs, …)
- construction rules
  - define admissible sentence structures
  - avoid ambiguous or imprecise constructions
- interpretation rules
  - control logical analysis of admissible sentences
  - resolve remaining ambiguities
- style guide

ACE Vocabulary: Function Words & Fixed Phrases

- function words and some fixed phrases are predefined and cannot be changed by users
- predefined function words
  - determiners, quantifiers, prepositions, coordinators, negation words, pronouns, query words, copula *be,* Saxon genitive marker *'s*
  - natural numbers
- predefined fixed phrases
  - *there is/are … such that*
  - *it is not the case that …*
  - *…*

ACE Vocabulary: Content Words

- contents words
  - nouns
  - verbs
  - adjectives
  - adverbs
- full-form common lexicon of close to 100'000 entries
- users can import domain-specific lexicons of content words that possibly override entries of the common lexicon
- words can be simple (*code*), or compound with (*check-code*) or without hyphen (*check code*)
- content words can have aliases, for instance abbreviations
- users can temporarily introduce missing content words by prefixing them with their respective word class

  A a:trusted man a:deliberately v:backs-up the n:web-page of the n:pizza-delivery-service.

Construction Rules: Noun Phrases

- singular countable noun phrases: *a/the/1 card*, *no card*, *every/each card*, *not every/each card, for every/each card*
- plural countable noun phrases: *the cards*, *some cards*, *3 cards*
- mass noun phrases: *some water*, *no water*, *all water*, *not all water*, *for all water*
- proper names: *John*, *Mr Miller*
- (non-) reflexive (possessive) pronouns: *he/she/it/they, him/her/it/them, himself/herself/itself/themselves, his/her/its/their, his/her/its/their own*
- indefinite pronouns: *someone*, *somebody*, *something*, *no one*, *nobody*, *nothing*, (*not*) *everyone*, (*not*) *everybody*, (*not*) *everything*
- generalised quantifiers: *at least 2 cards*, *at most 2 cards*, *more than 10 cards*, *less than 3 cards*
- measurement noun phrases: *2 kg of apples, 3 cubicmeter of water*

Construction Rules: Plural Noun Phrases

- ACE plural noun phrases have a collective or a distributive reading
- collective reading is the default

  A clerk enters 2 cards.
- distributive reading is indicated by *each of*

  A clerk enters each of 2 cards.
- noun phrase conjunction gives a plural object

  (each of) a customer and a clerk

Construction Rules: Modifying Noun Phrases

- adjective: *a rich customer, some cold water*
- adjective conjunction: *a rich and famous customer*
- relative phrase: *a customer who is rich*
- relative phrase conjunction: *a customer who is rich and who is famous*
- relative phrase disjunction: *a customer who is rich or who is famous*
- *of*-prepositional phrase: *a customer of John*
- Saxon genitive: *John's customer*
- possessive pronoun: *his (own) card*
- quoted string as apposition: *a message "John's fault"*
- variable as apposition: *a customer X*

  (NB: Variables introduced as appositions can be used anaphorically as noun phrases, e.g. *A customer X waits*. *X is tired*.)

Construction Rules: Verb Phrases

- intransitive (*wait*), transitive (*enter something, wait for something*), and ditransitive verbs (*give something to somebody, give somebody something*)
- third person singular/plural, indicative, simple present tense, active
- no modality (*can, must*), no intentionality (*believe*)
- prepositions of prepositional verbs and phrasal particles of phrasal verbs must be hyphenated to the verb (*wait-on*, *look-up*, *apply-for*)

Construction Rules: Verb Phrases

- copula *is*/*are* plus
  - noun phrase: *John is a rich customer*.
  - adjective: *John's wealth is enormous*.
  - comparative adjective: *John is richer than Mary*.
  - transitive adjective: *John is interested-in Mary and fond-of Bill*.
  - prepositional phrase: *John is in his own office*.

Construction Rules: Modifying Verb Phrases

- adverbs follow the verb ...
  A customer waits patiently.
  ... or – if present – its complements ...
  A customer inserts a card manually.
  ... or alternatively precede the verb
  A customer manually inserts a card.
- conjoined adverbs
  A customer inserts a card carefully and manually.
- (conjoined) prepositional phrases
  A customer inserts a card in the bank at a time T.
- combination of adverbs and prepositional phrases
  A customer inserts a card carefully into the slot.
  alternatively
  A customer carefully inserts a card into the slot.

Construction Rules: Modifying Verb Phrases

- notice the difference between a prepositional/phrasal verb and a verb with a prepositional phrase

  A steward waits-on a table.

  vs.

  The food waits on the table.

  A student is interested-in a course.

  vs.

  A student is interested in a classroom.

Construction Rules: Verb Phrase Coordination

- verb phrases can be coordinated by *and* and *or*
- verb phrase conjunction

  A screen flashes and blinks.
- verb phrase disjunction

  A screen flashes or blinks.
- combinations of conjunctions and disjunctions follow standard binding order of conjunction and disjunction

  A screen {flashes and blinks} or is dark.
- overriding the binding order by commas

  A screen flashes, and {blinks or is dark}.
- *and* and *or* have only a logical, no temporal meaning

Construction Rules: ACE Texts

- an ACE text is a sequence of anaphorically interrelated declarative sentences
- declarative sentences
  - simple sentences
  - composite sentences

- interrogative sentences query the contents of ACE texts

Construction Rules: Simple Sentences

- simple sentences have the structure

  subject + predicate + complements + adjuncts

  complements are the direct and indirect objects

  adjuncts are optional adverbs and prepositional phrases

- examples

  A customer waits.

  A customer inserts a card.

  A customer gives a card to a clerk.

  (alternatively: *A customer gives a clerk a card*.)

  A customer inserts a card manually into a slot.

Construction Rules: Simple Sentences

- it is possible to create well-formed simple sentences without a verb using the *there is*/*are* construct that introduces only an object custome

  There is a customer.

- no adjuncts or complements

  **There is a customer in the bank*.

- relative phrases are possible

  There is a customer who waits.

Construction Rules: Composite Sentences

- composite sentences are recursively built from simpler sentences with the help of the predefined constructors
  - coordination
  - quantification
  - negation
  - subordination
- example

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

Construction Rules: Coordination

- sentences can be coordinated by *and* and *or*
- sentence conjunction

  The screen blinks and John waits.
- sentence disjunction

  The screen blinks or John waits.
- standard binding order of conjunction and disjunction
- overriding of standard binding order by commas
- *and* and *or* have only a logical, no temporal meaning

Construction Rules: Quantification

- existential quantification
  There is a card. There is some water.
  John enters a card. John drinks some water.
- universal quantification
  John enters every card.
- global existential quantification
  There is a code that every clerk enters.
  or equivalently
  There is a code such that every clerk enters it.
- global universal quantification
  For every code (there is) a clerk (such that he) enters it.

Construction Rules: Negation

- negated existential quantifier

  John enters no code.
- negated universal quantifier

  John enters not every code.
- negated generalised quantifiers

  John enters not more than 2 cards.
- verb phrase negation

  John does not enter a code.
- negated copula

  Some water is not drinkable.
- sentence negation

  It is not the case that a screen blinks.

  It is not the case that a screen blinks and that the computer sleeps.

Construction Rules: Subordination

- ACE knows two forms of subordination: relative phrases (see 'Modifying Noun Phrases') and conditional sentences
- conditional sentences are built with the help of *if … then*

  If John enters a card then the automated teller accepts it.
- equivalence of universally quantified and conditional sentences

  Every customer enters a card.

  is equivalent to

  If there is a customer then the customer enters a card.

Construction Rules: Interrogative Sentences

- ACE also allows two forms of interrogative sentence
  - yes/no queries
  - wh-queries
- yes/no queries

  Does John enter a card?

  Is the card valid?
- wh-queries

  Who enters what?

  Which customer enters a card?

  Where does John enter a card?

  When does John enter a card?

  How does John enter a card?
- queries can consist of any number of declarative sentences followed by one interrogative sentence; this can be used to structure complex queries or to temporarily introduce additional knowledge

Constraining Ambiguity: Structural Ambiguity

- to constrain the ubiquitous structural ambiguity of natural language ACE employs three simple means
  - some ambiguous constructs are not part of ACE; unambiguous alternatives are available in their place
  - all remaining ambiguous constructs are interpreted deterministically on the basis of a small set of *interpretation rules*
  - users can accept the assigned interpretation, or they must rephrase the input to obtain another one
- ACE sentences are not ambiguous; however the same sentences can be ambiguous in full English

Interpretation Rules: Ambiguity

- 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 carries a code} and opens an account.
- to express coordination within the relative clause the relative pronoun has to be repeated

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

Interpretation Rules: Ambiguity

- the scope of sentence negation *it is not the case that* extends to the end of a simple sentence

  {It is not the case that a man waits} and a dog barks.
- to express coordination within the scope of sentence negation the word *that* has to be repeated

  {It is not the case that a man waits and that a dog barks}.
- in *if-then*-sentences the scope of the *if*-part and the scope of the *then*-part extend to the end of a coordination

  {If a man waits and a dog barks} then {a woman smiles and a cat sleeps}.

Interpretation Rules: Ambiguity

- if an adverb can modify the preceding or the following verb then it refers to the preceding verb

  A customer who {enters a card <u>manually</u>} types a code.

- the textual position of a quantifier opens its scope that extends to the end of the sentence, or in a coordination to the end of the respective coordinated phrase

  <u>A customer</u> types <u>every code</u>. $\therefore[$

  <u>Every customer</u> types <u>a code</u>. $[\therefore$

Constraining Ambiguity: Plural Noun Phrases

- plural noun phrases are highly ambiguous
- of the many readings of plural noun phrases ACE provides only the collective and the distribute readings
- collective reading is the default

  A clerk enters 2 cards.
- distributive reading is indicated by *each of*

  A clerk enters each of 2 cards.

Constraining Ambiguity: Lexical Ambiguity

- in general, controlling lexical ambiguity is the responsibility of the users
- some nouns occur both as countable and as mass; this lexical ambiguity is easily resolved by the correct use of determiners
  - countable noun: *a/the/every laundry*, *the laundries*, *some laundries, …*
  - mass noun: *some laundry*, *all laundry*, …
- further lexical ambiguity between countable and mass nouns is resolved by the parser preferring the countable interpretation; the mass interpretation can be enforced using the *of*-propositional phrase
  - countable: *John's laundry* — mass: *some laundry of John*
  - countable: *her laundry* — mass: *some laundry of her*
  - countable: *no laundry* — mass: *some laundry does/is not, it is not the case that some laundry*

Constraining Ambiguity: Lexical Ambiguity

- verbs are highly ambiguous, since the same verb can appear in the categories intransitive, transitive and ditransitive, and furthermore can occur with and without phrasal particles and prepositions as integral constituents
- to constrain this type of lexical ambiguity ACE expects that the phrasal particle of a phrasal verb (look up, drop out, shut down) and the preposition of a prepositional verb (look at, apply for) are hyphenated to the verb
  - *A steward waits-on the table.* (vs. *The food waits on the table*.)
  - *John looks-up an entry.* (vs. *John looks up the alley*.)
  - *What does John apply-for?* (vs. *John applies for the second time*.)

Constraining Ambiguity: Lexical Ambiguity

- hyphenation does not apply to ditransitive verbs since the prepositional complement is not adjacent to the verb and does not easily lead to ambiguity
  - *John gives a card to a clerk.*
  - *Who does John give a card to?*
- hyphenation can lead to ACE constructs not acceptable in full English
  - *There is an entry. John looks-up it.*

  that can easily be avoided using a definite noun phrase or a variable instead of a pronoun to express the anaphoric reference
  - *There is an entry. John looks-up the entry.*
  - *There is an entry E. John looks-up (the entry) E.*

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?

Anaphoric References

- ACE texts are interrelated by anaphoric references, i.e. references to textually preceding noun phrases
- anaphoric references can be
  - proper names: *John*
  - pronouns: *a card* Λ *it*, *itself*
  - definite noun phrases: *a card, some water* Λ *the card, the water*
  - variables: *a card X* Λ *the card X*, *X*
- example

  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.

Interpretation Rules: Anaphoric References

- proper names like *John* or *Mr Miller* always denote the same object and thus serve as their own anaphoric references
- in all other cases resolution of anaphoric references is governed by accessibility, recency, specificity, and reflexivity

Interpretation Rules: Accessibility of Noun Phrases

- accessibility: a noun phrase is not accessible if it occurs in a negated sentence

  *John does not enter a card. *It is correct.*

- accessibility: a noun phrase is not accessible if it occurs in a conditional sentence

  *Every customer has a card. *It is correct.*

  (use instead: *Every customer has a card that is correct.*)

  but a noun phrase in the *if*-part of a conditional sentence is accessible in the *then*-part

  If a customer has a card then he enters it.

- accessibility: a noun phrase in a disjunction is only accessible in subsequent disjuncts

  *A customer enters a card or drops it. *It is dirty.*

Interpretation Rules: Anaphoric References

- If the anaphor is a non-reflexive personal pronoun (*he*, *him*, …) or a non-reflexive possessive pronoun (*his*, …) then the anaphor is resolved with the most recent accessible noun phrase that agrees in gender and number, and that is not the subject of the sentence.
- examples

  John has a card. Bob sees him and takes it.

  **John sees his wife.* (use *John sees his own wife.*)

Interpretation Rules: Anaphoric References

- If the anaphor is a reflexive personal pronoun (*herself*, …) or a reflexive possessive pronoun (*her own*, …) then the anaphor is resolved with the subject of the sentence in which the anaphor occurs if the subject agrees in gender and number with the anaphor.
- example

  Mary takes her own card and gets some money for herself.

Interpretation Rules: Anaphoric References

- If the anaphor is a definite noun phrase then it is resolved with the most recent and most specific accessible noun phrase that agrees in gender and number.
- example

  There is a blue ball. There is a red ball. John sees the ball. Mary sees the blue ball.
- If a definite noun phrase cannot be resolved then it is interpreted as an indefinite noun phrase introducing a new object.

Interpretation Rules: Anaphoric References

- If the anaphor is a variable then it is resolved with an accessible noun phrase that has the variable as apposition.
- example

  John has a card X and a card Y. Mary takes the card. Bob takes the card X. Harry takes Y.
- example: predecessor is not accessible

  If a customer has a card C then the customer enters C.
  **C is not valid.*

Very Brief Style Guide

- While the ACE parser will unravel any syntactically correct sentence, however complex, *you* may have problems to do so.
- Thus
  - avoid unnecessarily complex constructs involving coordination, quantification and negation
  - use several simpler sentences instead of a complex one connecting these sentences by anaphoric references
  - if possible avoid plural and replace it by singular
  - recall the construction and interpretation rules
  - recall the constraints of anaphoric references
- Remember that your text is the only source of information; there is no hidden knowledge.

Applications of ACE

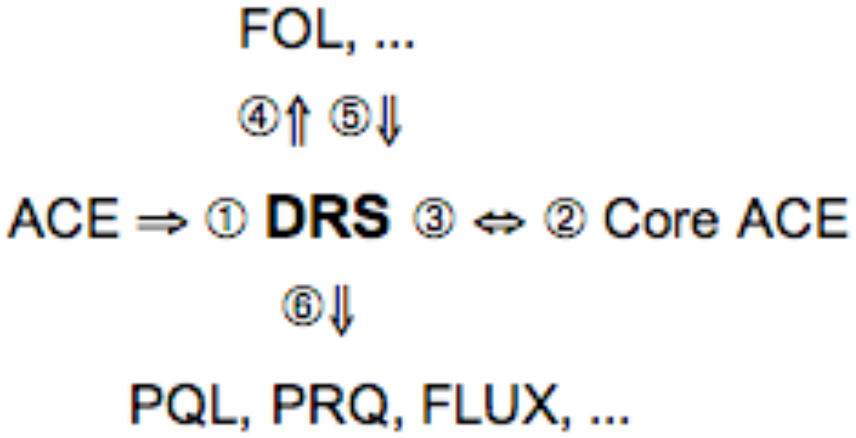
- *specifications*: automated teller machine, Kemmerer's library data base, Schubert's Steamroller, data base integrity constraints, Kowalski's subway regulations etc.
- *natural language interfaces*: model generator EP Tableaux (Munich), FLUX agent/robot control (Dresden), MIT's process query language (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 (business and policy rules, protein ontology etc.)

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 finds all minimal inconsistent subsets
  - if an ACE text is consistent then RACE finds its minimal model
  - if an ACE text is inconsistent then RACE finds all inconsistent subsets
  - if a query can be answered on the basis of an ACE text then RACE finds all answers
- RACE provides a proof justification in ACE
- RACE uses first-order axioms and Prolog axioms to reason about plurals and natural numbers
- RACE demo at www.ifi.unizh.ch/attempto

Specific Application: Semantic Web

- complement the formal languages of the semantic web with the familiarity of natural language
- translate ACE into languages of the semantic web: UML, OWL, ...
- verbalise languages of the semantic web in ACE using the DRS as interlingua



From the "Maltese Collection"

- ACE version of a query from "Web and Semantic Web Query Languages: A Survey" (Bailey et al.)
  - original: *Select all data items with any relation to the book titled "Bellum Civile".*
  - ACE: *Somebody selects every n:data-item that is a:related to the book that has the title "Bellum Civile".*
- ACE version of a rule from the Case Study "Financial Trading" (Tabet &Wagner)
  - original: *No securities issued by a restricted user must be bought.*
  - ACE: *Nobody buys the securities that a a:restricted user issues.*

Conclusions

- 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/