

Attempto Controlled English for Knowledge Representation

Norbert E. Fuchs, Kaarel Kaljurand, Tobias Kuhn

Department of Informatics & Institute of Computational Linguistics
University of Zurich

{fuchs, kalju, tkuhn}@ifi.uzh.ch

<http://attempto.ifi.uzh.ch/site/>

University of Bonn 2008

Overview

- Languages for Knowledge Representation
- Attempto Controlled English (ACE)
- Language ACE
- Translating ACE into First-Order Logic
- Attempto Tools
- Typical Applications of ACE
- Hands-On Training

Problem Knowledge Representation

- How should one represent the biological fact that every protein has a terminus?
- questions to be considered
 - 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 the biological fact that every protein has a terminus
- solutions: example representations

| | |
|-------------------|---|
| first-order logic | $\forall X(\text{protein}(X) \rightarrow \exists Y(\text{terminus}(Y) \wedge \text{has}(X, Y)))$ |
| DL | $\text{Protein} \sqsubseteq \exists \text{has}.\text{Terminus}$ |
| OWL (RDF/XML) | <pre> <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> </pre> |
| UML | <pre> classDiagram class Protein class Terminus Protein "1..*" *-- Terminus </pre> |
| 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 "Card accepted" and C is satisfied. No card that does not have a correct code is accepted. It is false 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 language ACE and the ACE parser do 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 ACE parser as uninterpreted syntactic elements, i.e. any meaning of these words is solely added by the human writer or reader.

The Language ACE

- like every language, ACE has
 - vocabulary/morphology
 - syntax
 - semantics
 - pragmatics
- ... and one must learn it (but wait until later)
- if you know English then you already know most of ACE
- thus let's see how ACE differs from English

English and ACE

- English is both syntactically and semantically more powerful than ACE
- English is used in human-human communication while ACE is also meant to be used in human-computer communication...
- ... which means that ACE is really a formal language (like various logics)
- ACE simply resembles English syntactically and semantically

English and ACE

- like English, ACE allows syntactically different sentences to express the same meaning (i.e. there is a lot of synonymy)
- unlike English, each ACE text is interpreted in just one way
 - ACE is not ambiguous
 - an ACE text read as an English text can have other meanings

Learning ACE

- vocabulary/morphology
 - predefined *function words* (articles, conjunctions, ...)
 - predefined *fixed phrases* ('there is a ...', 'it is false that ...')
 - user-defined *content words* (nouns, verbs, adjectives, adverbs)
- construction rules (syntax)
 - define admissible sentence structures
 - avoid ambiguous or imprecise constructions
- interpretation rules (semantics)
 - control logical analysis of admissible sentences
 - resolve remaining ambiguities
- style guide, tools (pragmatics)

Vocabulary

- content words
 - dog, cat, like, red, manually, ...
- content words can change morphologically
 - a man, 2 men
 - likes, does not like, is liked by
 - good, better than, best
- function words
 - if, then, who, 42, and, or, not, ...

ACE Content Words

- ACE content words are *nouns, verbs, adjectives* and *adverbs*
- set of ACE content words is infinitely large and dynamic
- all English content words are also ACE content words
 - e.g. podcast, groovy
- multiword units are always hyphenated
 - e.g. fill-in
 - e.g. persona-non-grata
 - e.g. switch-off

Semantics of Content Words

- no predefined lexical semantics
 - e.g. *bank* is underspecified
 - e.g. *narcissist* is just a noun with no relation to the verb *like*
 - e.g. *unacceptable* is just an adjective with no encoded negation
- users can of course add semantics by using ACE sentences
 - Every narcissist likes himself.
 - Everything that is not acceptable is unacceptable.
- this information is not used during parsing, but only by an eventual reasoning tool

ACE Nouns

- common nouns
 - countable: man, woman, cat
 - mass: water, money
- proper names
 - John, Mary
- countable nouns and proper names distinguish singular and plural
- all nouns carry gender information (masculine, feminine, neuter)
- measurement nouns
 - basic SI units: m, kg, ...

ACE Verbs

- subcategorisation (intransitive, transitive, ditransitive)
 - sit, like, give
- two forms of ditransitives
 - give something to somebody, give somebody something
- singular and plural forms
 - likes, like
- past participle forms for transitive and ditransitive verbs
 - liked
 - given
- phrasal and prepositional verbs
 - drop-out, fill-in

ACE Adjectives and Adverbs

- positive, comparative and superlative adjectives
 - tall, taller than somebody, tallest
- adjectives can have a PP-object
 - fond-of something
- positive, comparative and superlative adverbs
 - quickly, more quickly, most quickly

Content Words: Implementation

- full-form common lexicon of close to 100'000 entries
- users can import domain-specific lexicons of content words that can override entries of the common lexicon
- 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.*
- ACE parser can in many cases guess the correct word class of an unknown content word on the basis of its context

ACE Function Words

- predefined function words
 - determiners, quantifiers, prepositions, coordinators, negation words, pronouns, query words, copula *be*, Saxon genitive marker 's
 - numbers
- predefined fixed phrases
 - *there is/are ... such that*
 - *it is false that ...*
- set of function words is limited and unchanging
- not all English function words are in ACE
 - e.g. hence, whom

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 two cards, more than 10 cards, less than three cards*
- measurement noun phrases: *2 kg of apples, 3 l of water*

Construction Rules: Complete and Unmodifiable Noun Phrases

- some elements of ACE are considered complete noun phrases that cannot be modified by, for instance, adjectives or relative phrases
- numbers
 - *2, three, 3.14, -3*
- strings
 - *"Go!"*
- sets
 - *{John, Mary}*
- lists
 - *[1, 2, 3]*

Construction Rules: Complete and Unmodifiable Noun Phrases

- variables
 - $X, X13$
 - variables are introduced
 - in apposition to noun phrases *a man X*
 - as "bare" variables $X (= something X)$
- expressions
 - $3*(X+2)$
 - "*abc*" & "*123*"
 - expressions are not evaluated by the parser

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.
- NP 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*
- *of*-prepositional phrase: *a customer of John*
- Saxon genitive: *John's customer*
- possessive pronoun: *his (own) card*
- 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 :

Relative Clauses

- relative clause: *a customer who knows John*
- relative clause (with inversion): *a customer who John knows*
- complex relative clauses
 - conjunction: *a customer who is rich and who is famous*
 - disjunction: *a customer who is rich or who is famous*
 - embedding: *a customer who sees a man who knows John*
 - embedding (with inversion): *a customer who a man who knows John sees*

Construction Rules: Verb Phrases

- intransitive (*wait*), transitive (*enter something, wait-for something*), and ditransitive verbs (*give something to somebody, give somebody something*)
- 3rd person singular/plural, present tense, active/passive
- modality (*can, must*)
- intentionality (*believe that*)
- 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 or – if present – its complements
 - *A customer waits patiently.*
 - *A customer inserts a card manually.*
- adverbs can also precede the verb
 - *A customer manually inserts a card.*
- adverbs can be conjoined (but not disjoined)
 - *A customer inserts a card carefully and manually.*
- prepositional phrases can be concatenated
 - *A customer inserts a card in the bank at a time T.*
- adverbs and prepositional phrases can be concatenated
 - *A customer inserts a card carefully into the slot.*
 - *A customer carefully inserts a card into the slot.*

Construction Rules: Complement vs Adjunct

- 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

- VPs can be coordinated by *and* and *or*
- conjunction
 - *A screen flashes **and** blinks.*
- 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.*
- order can be overridden by commas
 - *A screen flashes, **and** {blinks **or** is dark}.*
- NB: The brackets `{ }` are not part of ACE and are only used here to make the binding order explicit.

Construction Rules: ACE Texts

- ACE text is a sequence of anaphorically interrelated declarative sentences optionally followed by one interrogative sentence.
- declarative sentences
 - end with full stop
 - can be simple or composite
- interrogative sentences
 - end with a question mark
 - query the contents of ACE texts
- furthermore there are imperative sentences

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 card is inserted by a customer.* (*)
 - *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.*

(*) These two sentences have the same meaning.

Construction Rules:

there is Sentences


- it is possible to create well-formed simple sentences without a verb by using the *there is/are* construct that introduces only an object
 - *There is a customer.*
- no adjuncts or complements are allowed (because there is no main verb)
 - **There is a customer in the bank.*
 - NB: The asterisk * means here that the sentence is syntactically incorrect.
- relative clauses are possible (because a noun is present)
 - *There is a customer who waits.*

Construction Rules: Formulas

- logical formulas constitute another form of simple sentences
 - $10 = 4 + 6.$
 - $5 > 3.$
 - $X \geq 13.4.$
- formulas are not evaluated by the parser

ACE Tools

Attempto Project
News
People
Tools
APE (ACE parser)
RACE (ACE reasoner)
AceRules
AceRules (technical)
AceWiki
ACE View
OWL verbalizer
Documentation
Publications
Downloads
Talks
Courses
Cooperations
Mailing List
Contact



Attempto Tools

APE (ACE parser)

The ACE parser APE can be used through [APE Webclient](#), or directly, via a [APE Webservice](#).

Related documentation:

- [APE Webclient Help](#)
- [APE Webservice Specification](#)
- [ACE Lexicon Specification](#)

RACE (ACE reasoner)

The [ACE reasoner RACE](#) allows users to do deduction on ACE texts, for example consistency checks and query answering.

AceRules

AceRules is a forward-chaining rule system based on ACE, i.e. the rules and the answer sets are written in ACE. It supports multiple semantics (called 'modes'): courteous logic programs, stable models, and stable models with strong negation. There is a public webservice for AceRules. See the document [AceRules Webservice](#) for more information. Based on this webservice, there are two web interfaces: the [AceRules interface](#) as a demonstration for normal users and the [AceRules technical interface](#) for developers.

AceWiki

[AceWiki](#) is a semantic wiki using ACE. Unlike most other semantic wikis, the semantics are contained directly in the article texts and not in some form of annotations.

ACE View

[ACE View](#) is an ontology and rule editor that uses ACE in order to create, view and edit OWL 2 ontologies and SWRL rulesets.

OWL verbalizer

[OWL verbalizer](#) converts OWL 2 ontologies (expressed in XML Serialization) into ACE.

Externally developed tools

David Hirtle's [TRANSLATOR \(TRANSLator from LAnguage TO Rules\)](#) converts an ACE text into [RuleML](#). It is available as a Java Web Start application which uses the APE webservice as a backend to convert ACE texts into DRSs.

last updated: 24 Oct 2008 Kaarel Kaljurand

[back to top](#) | [home](#)

Attempto Parsing Engine (APE)

- for syntactically correct texts outputs the analysis of the text
 - tokens
 - syntax trees
 - paraphrase in ACE
 - various logical translations
- for erroneous texts
 - detects syntactic errors and unknown words in an ACE text
 - generates error and warning messages indicating the location and the possible causes of the errors, and suggesting remedies

Attempto Parsing Engine (APE)

- APE can be accessed
 - by a web-service (attempto.ifi.uzh.ch/site/docs/ape_webservice.html)
 - by a web-client (attempto.ifi.uzh.ch/ape/)
 - from Java programs (Attempto Java Packages downloadable under GNU LGPL from attempto.ifi.uzh.ch/site/downloads/)
- all APE interfaces are fully documented
- source code of APE plus some related tools is available under the GNU LGPL at attempto.ifi.uzh.ch/site/downloads/

Show Input text Paraphrase DRS DRS XML FOL PNF OWL FSS OWL RDF Tokens Syntax
Options Guess unknown words Do not use Clex
Lexicon Reload the lexicon from URL

If a talk ends then everybody can ask a question.

↑ ↓ Analyse

overall: 0.469 sec (tokenizer: 0.000 parser: 0.010 refres: 0.000) :: Thu Sep 04 2008 17:42:37 GMT+0200 (CEST)

If a talk ends then everybody can ask a question.

PARAPHRASE

If a talk ends then if there is somebody X1 then it is possible that X1 asks a question.

DRS

```
[ ]
  [A, B]
  object(A, talk, countable, na, eq, 1)-1
  predicate(B, end, A)-1
  =>
  [ ]
    [C]
    object(C, somebody, countable, na, eq, 1)-1
    =>
    [ ]
      <|>
      [D, E]
      object(D, question, countable, na, eq, 1)-1
      predicate(E, ask, C, D)-1
```

Hide menu Help

Show Input text Paraphrase DRS DRS XML FOL PNF OWL FSS OWL RDF Tokens Syntax
Options Guess unknown words Do not use Clex
Lexicon Reload the lexicon from URL

If a talk ends then everybody can aks a question.

↑ ↓ Analyse

overall: 0.086 sec (tokenizer: 0.010 parser: 0.010 refres: 0.000) :: Thu Sep 04 2008 17:45:18 GMT+0200 (CEST)

| | Type | Sentence | Problem | Suggestion |
|-------|----------|----------|--|---|
| error | sentence | 1 | If a talk ends then everybody can <> aks a question. | This is the first sentence that was not ACE. The sign <> indicates the position where parsing failed. |
| error | word | | aks | ask |

If a talk ends then everybody can aks a question.

PARAPHRASE

NOT IMPLEMENTED

DRS

No conditions
[]

Hide menu Help

Show Input text Paraphrase DRS DRS XML FOL PNF OWL FSS OWL RDF Tokens Syntax
Options Guess unknown words Do not use Clex
Lexicon Reload the lexicon from URL

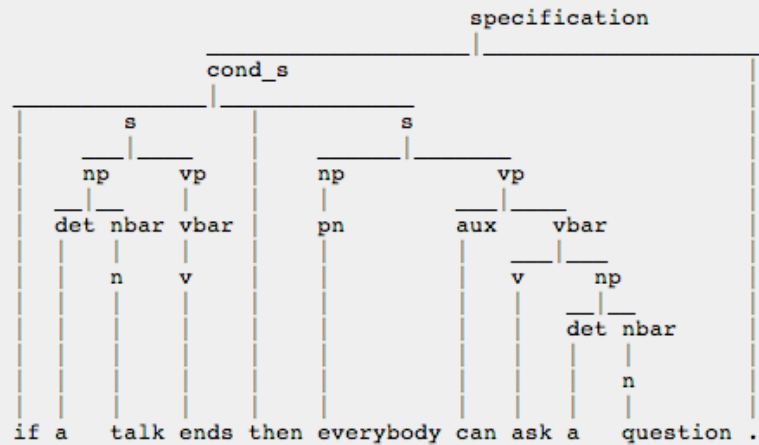
If a talk ends then everybody can ask a question.

↑ ↓ Analyse

overall: 0.137 sec (tokenizer: 0.010 parser: 0.010 refes: 0.000) :: Thu Sep 04 2008 20:38:27 GMT+0200 (CEST)

If a talk ends then everybody can ask a question.

SYNTAX



Hands-On Training

- open APE (ACE Parser): go to attempto.ifi.uzh.ch/site/tools/ or directly to attempto.ifi.uzh.ch/ape/
- read the APE help file and have a look at the menu
- try the example sentences provided by the web-client
(hint: use the ↑ and ↓ buttons to scroll through the sentences)
- use various menu settings and check the output
- introduce intentional errors into the sentences

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.*
 - *$3 < 4$ and $3 = < 5$.*
- sentence disjunction
 - *The screen blinks **or** John waits.*
 - *$X < 4$ or $X > 10$.*
- overriding of standard binding order by commas
 - *The screen blinks **or** John waits, **and** Mary sleeps.*

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. Every card is valid.*
- global existential quantification
 - *There is a code that every clerk enters.*
 - or equivalently: *There is a code such that every clerk enters it.*
 - or equivalently: *There is a code that is entered by every clerk.*
- global universal quantification
 - *For every code a clerk enters it.*

Construction Rules: Negation

- negated existential quantifier
 - *John enters **no code**.*
- negated universal quantifier
 - *John enters **not every code**.*
- VP negation
 - *John **does not enter** a code.*
- negated copula
 - *Some water **is not** drinkable.*
- sentence negation
 - ***It is false that** a screen blinks.*
 - ***It is false that** a screen blinks **and that** the computer sleeps.*

Construction Rules: Subordination

- ACE knows several forms of subordination
 - relative phrases (we discussed them already when talking about noun phrase modification)
 - conditional sentences
 - sentence subordination
 - modality

Construction Rules: Conditional Sentences

- conditional sentences are built with the help of *if ... then*
 - *If John enters a card then the automatic 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: Sentence Subordination

- negation
 - *It is false that a customer inserts a card.*
- negation as failure (to support translation of ACE into rules and into languages like Prolog)
 - *It is not provable that a customer inserts a card.*
- sentence as an object of a verb
 - *A clerk believes that a customer inserts a card.*

Construction Rules: Modality

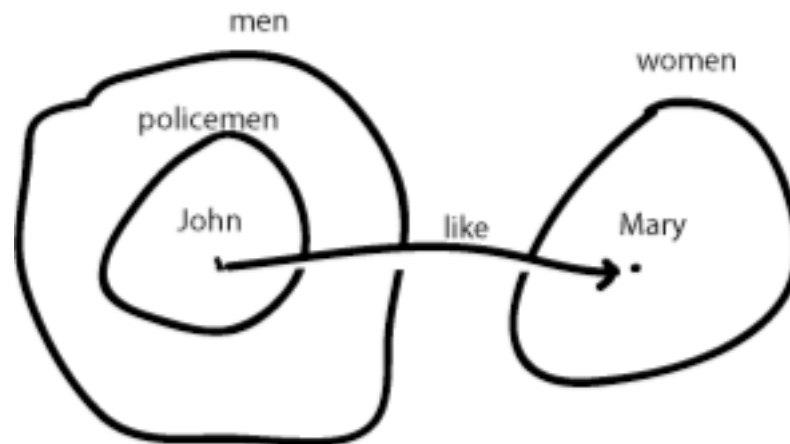
- possibility
 - *A trusted customer can insert a card.*
 - *A trusted customer cannot insert a card.*
 - *It is possible that a trusted customer inserts a card.*
 - *It is not possible that a trusted customer inserts a card.*
- necessity
 - *A trusted customer must insert a card.*
 - *A trusted customer does not have to insert a card.*
 - *It is necessary that a trusted customer inserts a card.*
 - *It is not necessary that a trusted customer inserts a card.*

Construction Rules: Interrogative Sentences

- ACE 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?*
 - *How does John enter a card?*

Hands-On Training

- express the following situation in ACE under the assumption that the diagram represents the complete knowledge



- express also what is not the case
- extend the situation in whatever way

Ambiguity in English and in ACE

- English is a highly ambiguous language
- ambiguity can occur on several levels
 - syntax: *A man sees a girl with a telescope. John has a flat mate.*
 - scope: *Everybody loves somebody.*
 - lexical: *Mary sees a bank.*
- English relies heavily on context to resolve ambiguity
 - *A man sees a girl with a green dress.*
- ACE uses only structural information to resolve ambiguity
 - *A man {sees a girl with a telescope.}*
 - *A man {sees a girl with a green dress.}*
- ACE sentences are not ambiguous; however the same sentences can be ambiguous when read as full English

Constraining Ambiguity: Structural Ambiguity

- ACE employs three simple means to constrain the ubiquitous structural ambiguity of natural language
 - 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
- here is an ...

Constraining Ambiguity: Structural Ambiguity

- ... example:
 - 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: 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

- scope of sentence negation *it is false that* extends to the end of a simple sentence
{It is false 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 false 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 manually} types a code.*
- 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
 - *{A customer types {every code}}.* $\exists \forall$
 - *{Every customer types {a code}}.* $\forall \exists$

Constraining Ambiguity: Plural Noun Phrases

- plural NPs are highly ambiguous
- of the many readings of plural NPs 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

- verbs are highly ambiguous, since the same verb can appear as 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 **E**.*

Summary of ACE's 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 a 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 made by
 - proper names: *John*
 - pronouns: *it, itself*
 - definite noun phrases: *the card, the water, the red card, the man who waits*
 - variables: *the card X, X*
- *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
 - reflexivity

Interpretation Rules: Accessibility

- noun phrase is not accessible if it occurs in a negated sentence
 - *John does not enter a card. *It is correct.*
- 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.*
- noun phrase in a disjunction is only accessible in subsequent disjuncts
 - *A customer enters a card or drops it. *It is dirty.*

Interpretation Rules: Pronominal 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: Pronominal 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: Definite Noun Phrases

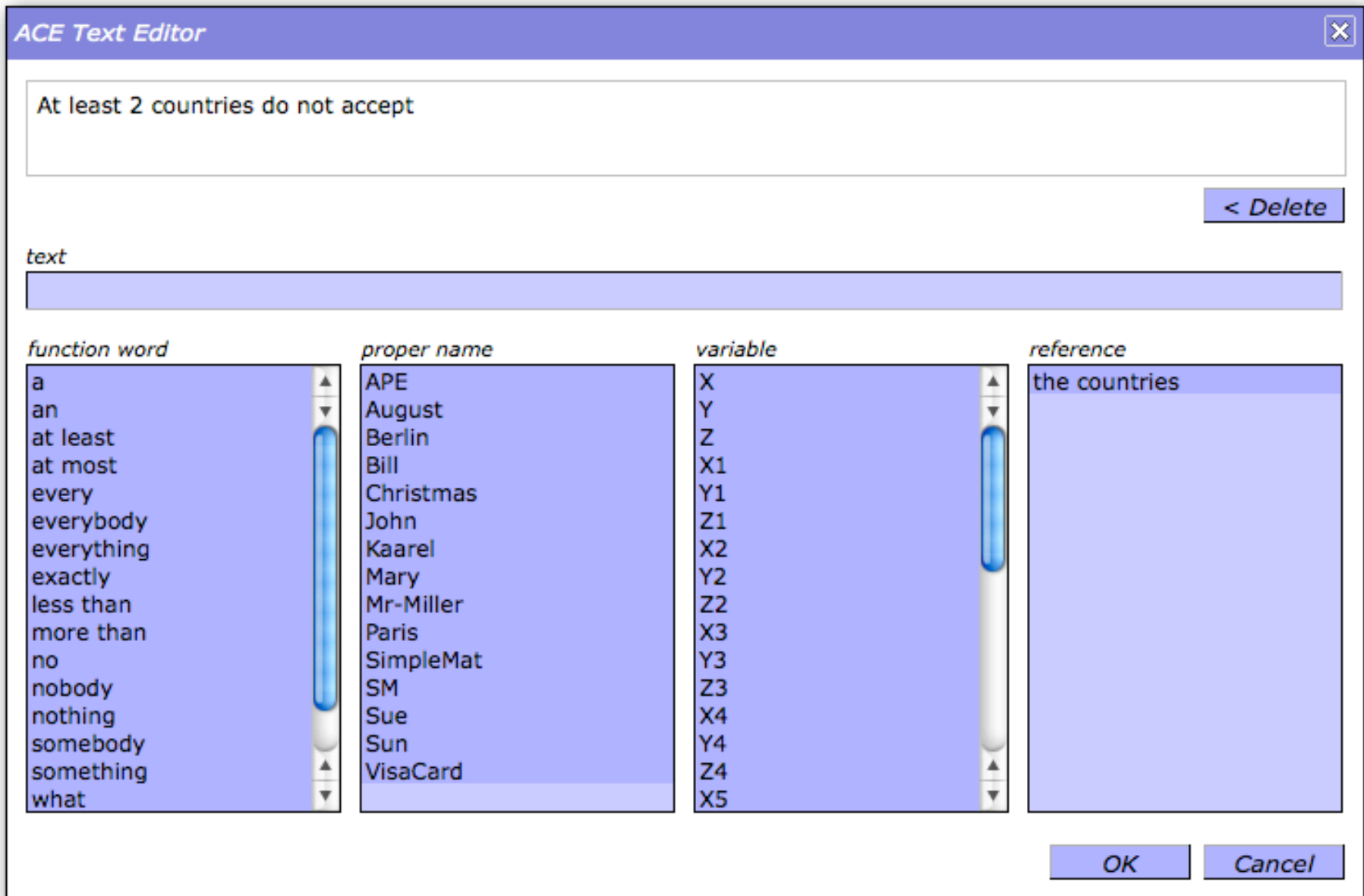
- if the anaphor is a definite NP then it is resolved with the most recent and most specific accessible noun phrase that agrees in gender and number
- example
 - *There is a blue ball. There is a red ball. John sees the ball. Mary sees the blue ball.*
- pragmatics often requires using a definite noun phrase that is not meant anaphorically
 - if a definite NP cannot be resolved then it is interpreted as an indefinite noun phrase introducing a new object
 - *John goes to the bank. (= John goes to a bank.)*

Interpretation Rules: Variables

- if the anaphor is a variable then it is resolved with an accessible noun phrase that has the variable as apposition, or with a previously introduced "bare" variable
- example
 - *John has a card X and a card Y. Mary takes the card. Bob takes the card X. Harry takes Y.*
 - *John has X. X is not described.*
- example: predecessor is not accessible
 - *If a customer has a card C then the customer enters C. C is not valid.*
 - in this case the second occurrence of C introduces a new (bare) variable C

ACE Editor

- APE requires users to learn and to recall the ACE construction and interpretation rules
- ACE Editor is an experimental predictive editor that helps users to construct syntactically and lexically correct ACE texts by just clicking on words and word classes
- alternatively users can freely input ACE texts
- ACE Editor can be accessed via a web-client (attempto.ifi.uzh.ch/aceeditor/)



Hands-On Training

- go to the ACE Editor at attempto.ifi.uzh.ch/aceeditor/
- enter ACE sentences by clicking
- enter ACE sentences by clicking and free input

From ACE to First-Order Logic

- input: ACE text
- target: Extended Discourse Representation Structure (DRS)
 - uses syntactic variant of language of standard first-order logic
 - internal representation as term $drs(Referents, Conditions)$ where *Referents* is a set of quantified variables and *Conditions* a set of logical conditions for *Referents*
- 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

(Pretty-Printed APE Output)

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

PARAPHRASE

If a company X1 buys at least 2 standard machines then the company X1 gets a discount.

DRS

```
[ ]
  [A, B, C]
  object(A, company, countable, na, eq, 1)-1
  object(B, machine, countable, na, geq, 2)-1
  property(B, standard, pos)-1
  predicate(C, buy, A, B)-1
  =>
  [D, E]
  object(D, discount, countable, na, eq, 1)-1
  predicate(E, 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, company, countable, na, eq, 1)-1

object(B, machine, countable, na, geq, 2)-1

property(B, standard, pos)-1

predicate(C, buy, A, B)-1

=>

[D, E]

object(D, discount, countable, na, eq, 1)-1

predicate(E, get, A, D)-1

Properties of DRS Representation

Only Predefined Relation Symbols

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

[]

[A, B, C]

object(A, company, countable, na, eq, 1)-1

object(B, machine, countable, na, geq, 2)-1

property(B, standard, pos)-1

predicate(C, buy, A, B)-1

=>

[D, E]

object(D, discount, countable, na, eq, 1)-1

predicate(E, get, A, D)-1

Properties of DRS Representation

Predicates as Arguments

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

[]

[A, B, C]

object(A, **company**, countable, na, eq, 1)-1

object(B, **machine**, countable, na, geq, 2)-1

property(B, **standard**, pos)-1

predicate(C, **buy**, A, B)-1

=>

[D, E]

object(D, **discount**, countable, na, eq, 1)-1

predicate(E, **get**, A, D)-1

Properties of DRS Representation

Quantity Information

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

[]

[A, B, C]

object(A, company, countable, na, eq, 1)-1

object(B, machine, countable, na, geq, 2)-1

property(B, standard, pos)-1

predicate(C, buy, A, B)-1

=>

[D, E]

object(D, discount, countable, na, eq, 1)-1

predicate(E, get, A, D)-1

Properties of DRS Representation

Indices for Tracking

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

[]

[A, B, C]

object(A, company, countable, na, eq, 1)-1

object(B, machine, countable, na, geq, 2)-1

property(B, standard, pos)-1

predicate(C, buy, A, B)-1

=>

[D, E]

object(D, discount, countable, na, eq, 1)-1

predicate(E, get, A, D)-1

Summary of DRS Representation

- advantages of DRS representation
 - DRS: integrate discourse anaphora
 - first-order: eases automated deduction and reusability
 - reification: possible quantification over predicates in first-order logic
 - plurals: represent plurals in first-order logic
- DRSs have been translated into other first-order languages
- via DRSs tools can make use of ACE as interface language

Relevant Documentation

- *ACE in a Nutshell* is a short overview of the ACE language
- *ACE Lexicon Specification* describes the allowed content words
- *ACE Construction Rules* lists the rules that determine which sentences belong to ACE
- *ACE Interpretation Rules* lists the rules that remove the ambiguity from the ACE sentences
- *ACE Troubleshooting Guide* describes how to use ACE, including how to avoid pitfalls
- *ACE Syntax Report* contains an abstract syntax of ACE
- *DRS Report* describes the DRS language
- this and more documentation is found at attempto.ifi.uzh.ch/site/docs/

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

Other Controlled Languages

- http://en.wikipedia.org/wiki/Controlled_natural_language
- research
 - Schwitter (Macquarie): Processable English (PENG)
 - Sowa (VivoMind): Common Logic Controlled English (CLCE)
 - Pratt-Hartmann (Manchester): E2V
 - Dolbear et al. (Ordnance Survey): Rabbit
 - ...
- industry
 - Clark et al (Boeing): Computer-Processable Language (CPL)
 - ...

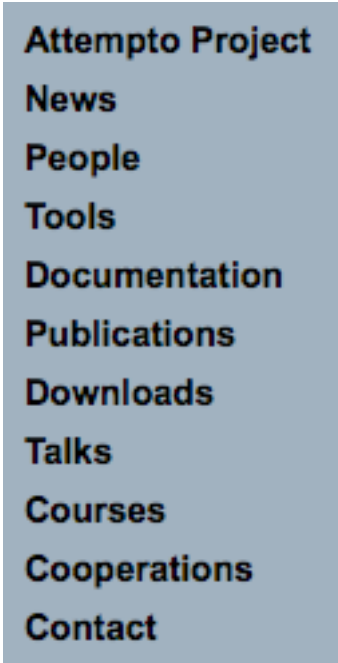
Applications of ACE

- *specifications that we developed*: 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)
- *medicine*: reports, hospital guidelines (Yale)
- *semantic web*: business & policy rules, translation into and from web-languages, protein ontology (EU Network of Excellence REVERSE)
- annotations of web-pages in controlled natural language (Macquarie)

Attempto Web-site

<http://attempto.ifi.uzh.ch/site/>

- Attempto web-site answers questions you may have concerning ...



Attempto Project
News
People
Tools
Documentation
Publications
Downloads
Talks
Courses
Cooperations
Contact

- here you will soon find the slides of this course

Hands-On Training

- Paraphrase the following text in ACE and submit it to the parser. If words are not found in the common lexicon prefix them with p: (for proper names), n: (for nouns) v: (for verbs), and a: (for adjectives and adverbs). Hyphenate multiwords and phrasal and prepositional verbs.

Attempto Controlled English (ACE) is a language specifically designed to write specifications. ACE is a controlled natural language, i.e. a subset of English with a domain specific vocabulary and a restricted grammar in the form of a small set of construction and interpretation rules. This means that all ACE sentences are correct English, but that not all English sentences are allowed in ACE. The restriction of full natural language to a controlled subset is essential for ACE to be suitable for knowledge representation purposes.

Hands-On Training

- problem
... is a language specifically **designed to write** specifications ...
- hint 1
replace passive by active and introduce an agent who designs ACE
- hint 2
use nominalisation, for instance for the writing, or sentence subordination, for instance (some transitive verb) that (some sentence)