Attempto Controlled English

From Wikipedia, the free encyclopedia

Attempto Controlled English (**ACE**) is a controlled natural language, i.e. a subset of standard English with a restricted syntax and a restricted semantics described by a small set of construction and interpretation rules.[1]

ACE can serve as knowledge representation, specification, and query language, and is intended for professionals who want to use formal notations and formal methods, but may not be familiar with them. Though ACE appears perfectly natural – it can be read and understood by any speaker of English – it is in fact a formal language.

ACE and its related tools have been used in the fields of software specifications, theorem proving, text summaries, ontologies, rules, querying, medical documentation and planning.

Here are some simple examples:

1. Every woman is a human.
2. A woman is a human.
3. A man tries-on a new tie. If the tie pleases his wife then the man buys it.

ACE construction rules require that each noun be introduced by a determiner (*a*, *every*, *no*, *some*, *at least 5*, ...). ACE interpretation rules decide that (1) is interpreted as universally quantified, while (2) is interpreted as existentially quantified. Sentences like "Women are human" do not follow ACE syntax and are consequently not valid.

Interpretation rules resolve the anaphoric references in (3): *the tie* and *it* of the second sentence refer to *a new tie* of the first sentence, while *his* and *the man* of the second sentence refer to *a man* of the first sentence. Thus an ACE text is a coherent entity of anaphorically linked sentences.

The Attempto Parsing Engine (APE) translates ACE texts unambiguously into discourse representation structures (DRS) that use a variant of the language of first-order logic.[2] A DRS can be further translated into other formal languages, for instance AceRules with various semantics,[3] OWL,[4] and SWRL. Translating an ACE text into (a fragment of) first-order logic allows users to reason about the text, for instance to verify, to validate, and to query it.

Contents

ACE in a nutshell

As an overview of the current version 6.6 of ACE this section:

- Briefly describes the vocabulary
- Gives an account of the syntax
- Summarises the handling of ambiguity
- Explains the processing of anaphoric references.

Vocabulary

The vocabulary of ACE comprises:

- Predefined function words (e.g. determiners, conjunctions)
- Predefined phrases (e.g. "it is false that ...", "it is possible that ...")
- Content words (e.g. nouns, verbs, adjectives, adverbs).

Grammar

The grammar of ACE defines and constrains the form and the meaning of ACE sentences and texts. ACE's grammar is expressed as a set of construction rules (http://attempto.ifi.uzh.ch/site/docs/ace_constructionrules.html). The meaning of sentences is described as a small set of interpretation rules (http://attempto.ifi.uzh.ch/site/docs/ace_interpretationrules.html). A Troubleshooting Guide (http://attempto.ifi.uzh.ch/site/docs/ace_troubleshooting.html) describes how to use ACE and how to avoid pitfalls.

ACE texts

An ACE text is a sequence of declarative sentences that can be anaphorically interrelated. Furthermore, ACE supports questions and commands.

Simple sentences

A simple sentence asserts that something is the case — a fact, an event, a state.

The temperature is -2 °C.
A customer inserts 2 cards.
A card and a code are valid.

Simple ACE sentences have the following general structure:

subject + verb + complements + adjuncts

Every sentence has a subject and a verb. Complements (direct and indirect objects) are necessary for transitive verbs (*insert something*) and ditransitive verbs (*give something to somebody*), whereas adjuncts (adverbs, prepositional phrases) are optional.

All elements of a simple sentence can be elaborated upon to describe the situation in more detail. To further specify the nouns *customer* and *card*, we could add adjectives:

A trusted customer inserts two valid cards.

possessive nouns and *of*-prepositional phrases:

John's customer inserts a card of Mary.

or variables as appositions:

John inserts a card A.

Other modifications of nouns are possible through relative sentences:

A customer who is trusted inserts a card that he owns.

which are described below since they make a sentence composite. We can also detail the insertion event, e.g. by adding an adverb:

A customer inserts some cards manually.

or, equivalently:

A customer manually inserts some cards.

or, by adding prepositional phrases:

A customer inserts some cards into a slot.

We can combine all of these elaborations to arrive at:

> John's customer who is trusted inserts a valid card of Mary manually into a slot A.

Composite sentences

Composite sentences are recursively built from simpler sentences through coordination, subordination, quantification, and negation. Note that ACE composite sentences overlap with what linguists call compound sentences and complex sentences.

Coordination

Coordination by *and* is possible between sentences and between phrases of the same syntactic type.

> A customer inserts a card and the machine checks the code.
> There is a customer who inserts a card and who enters a code.
> A customer inserts a card and enters a code.
> An old and trusted customer enters a card and a code.

Note that the coordination of the noun phrases *a card* and *a code* represents a plural object.

Coordination by *or* is possible between sentences, verb phrases, and relative clauses.

> A customer inserts a card or the machine checks the code.
> A customer inserts a card or enters a code.
> A customer owns a card that is invalid or that is damaged.

Coordination by *and* and *or* is governed by the standard binding order of logic, i.e. *and* binds stronger than *or*. Commas can be used to override the standard binding order. Thus the sentence:

> A customer inserts a VisaCard or inserts a MasterCard, and inserts a code.

means that the customer inserts a VisaCard and a code, or alternatively a MasterCard and a code.

Subordination

There are four constructs of subordination: relative sentences, *if-then* sentences, modality, and sentence subordination.

Relative sentences starting with *who*, *which*, and *that* allow to add detail to nouns:

> A customer who is trusted inserts a card that he owns.

With the help of *if-then* sentences we can specify conditional or hypothetical situations:

> If a card is valid then a customer inserts it.

Note the anaphoric reference via the pronoun *it* in the *then*-part to the noun phrase *a card* in the *if*-part.

Modality allows us to express possibility and necessity:

> A trusted customer can/must insert a card.
> It is possible/necessary that a trusted customer inserts a card.

Sentence subordination comes in various forms:

> It is true/false that a customer inserts a card.
> It is not provable that a customer inserts a card.
> A clerk believes that a customer inserts a card.

Quantification

Quantification allows us to speak about all objects of a certain class (universal quantification), or to denote explicitly the existence of at least one object of this class (existential quantification). The textual occurrence of a universal or existential quantifier opens its scope that extends to the end of the sentence, or in coordinations to the end of the respective coordinated sentence.

To express that all involved customers insert cards we can write

> Every customer inserts a card.

This sentence means that each customer inserts a card that may, or may not, be the same as the one inserted by another customer. To specify that all customers insert the same card — however unrealistic that situation seems — we can write:

> A card is inserted by every customer.

or, equivalently:

> There is a card that every customer inserts.

To state that every card is inserted by a customer we write:

Every card is inserted by a customer.

or, somewhat indirectly:

For every card there is a customer who inserts it.

Negation

Negation allows us to express that something is not the case:

A customer does not insert a card.
A card is not valid.

To negate something for all objects of a certain class one uses *no*:

No customer inserts more than 2 cards.

or, *there is no*:

There is no customer who inserts a card.

To negate a complete statement one uses sentence negation:

It is false that a customer inserts a card.

These forms of negation are logical negations, i.e. they state that something is provably not the case. Negation as failure states that a state of affairs cannot be proved, i.e. there is no information whether the state of affairs is the case or not.

It is not provable that a customer inserts a card.

Queries

ACE supports two forms of queries: *yes/no*-queries and *wh*-queries.

Yes/no-queries ask for the existence or non-existence of a specified situation. If we specified:

A customer inserts a card.

then we can ask:

Does a customer insert a card?

to get a positive answer. Note that interrogative sentences always end with a question mark.

With the help of *wh*-queries, i.e. queries with query words, we can interrogate a text for details of the specified situation. If we specified:

A trusted customer inserts a valid card manually in the morning in a bank.

we can ask for each element of the sentence with the exception of the verb.

Who inserts a card?
Which customer inserts a card?
What does a customer insert?
How does a customer insert a card?
When does a customer enter a card?
Where does a customer enter a card?

Queries can also be constructed by a sequence of declarative sentences followed by one interrogative sentence, for example:

There is a customer and there is a card that the customer enters. Does a customer enter a card?

Commands

ACE also supports commands. Some examples:

John, go to the bank!
John and Mary, wait!
Every dog, bark!
A brother of John, give a book to Mary!

A command always consists of a noun phrase (the addressee), followed by a comma, followed by an uncoordinated verb phrase. Furthermore, a command has to end with an exclamation mark.

Constraining ambiguity

To constrain the ambiguity of full natural language ACE employs three simple means:

- Some ambiguous constructs are not part of the language; unambiguous alternatives are available in their place

- All remaining ambiguous constructs are interpreted deterministically on the basis of a small number of interpretation rules
- Users can either accept the assigned interpretation, or they must rephrase the input to obtain another one.

Avoidance of ambiguity

In natural language, relative sentences combined with coordinations can introduce ambiguity:

> A customer inserts a card that is valid and opens an account.

In ACE the sentence has the unequivocal meaning that the customer opens an account, as reflected by the paraphrase:

> A card is valid. A customer inserts the card. The customer opens an account.

To express the alternative — though not very realistic — meaning that the card opens an account, the relative pronoun *that* must be repeated, thus yielding a coordination of relative sentences:

> A customer inserts a card that is valid and that opens an account.

This sentence is unambiguously equivalent in meaning to the paraphrase:

> A card is valid. The card opens an account. A customer inserts the card.

Interpretation rules

Not all ambiguities can be safely removed from ACE without rendering it artificial. To deterministically interpret otherwise syntactically correct ACE sentences we use a small set of interpretation rules. For example, if we write:

> A customer inserts a card with a code.

then *with a code* attaches to the verb *inserts*, but not to *a card*. However, this is probably not what we meant to say. To express that *the code* is associated with *the card* we can employ the interpretation rule that a relative sentence always modifies the immediately preceding noun phrase, and rephrase the input as:

> A customer inserts a card that carries a code.

yielding the paraphrase:

> A card carries a code. A customer inserts the card.

or — to specify that the customer inserts a card and a code — as:

> A customer inserts a card and a code.

Anaphoric references

Usually ACE texts consist of more than one sentence:

> A customer enters a card and a code. If a code is valid then SimpleMat accepts a card.

To express that all occurrences of card and code should mean the same card and the same code, ACE provides anaphoric references via the definite article:

> A customer enters a card and a code. If the code is valid then SimpleMat accepts the card.

During the processing of the ACE text, all anaphoric references are replaced by the most recent and most specific accessible noun phrase that agrees in gender and number. As an example of "most recent and most specific", suppose an ACE parser is given the sentence:

> A customer enters a red card and a blue card.

Then:

> The card is correct.

refers to the second card, while:

> The red card is correct.

refers to the first card.

Noun phrases within *if-then* sentences, universally quantified sentences, negations, modality, and subordinated sentences cannot be referred to anaphorically from subsequent sentences, i.e. such noun phrases are not "accessible" from the following text. Thus for each of the sentences:

> If a customer owns a card then he enters it.
> Every customer enters a card.
> A customer does not enter a card.
> A customer can enter a card.
> A clerk believes that a customer enters a card.

we cannot refer to *a card* with:

> The card is correct.

Anaphoric references are also possible via personal pronouns:

> A customer enters a card and a code. If it is valid then SimpleMat accepts the card.

or via variables:

> A customer enters a card X and a code Y. If Y is valid then SimpleMat accepts X.

Anaphoric references via definite articles and variables can be combined:

> A customer enters a card X and a code Y. If the code Y is valid then SimpleMat accepts the card X.

Note that proper names like *SimpleMat* always refer to the same object.

History

ACE has been under development in the Attempto project (http://attempto.ifi.uzh.ch) at the University of Zurich since 1995. In 2011, ACE version 6.6 was announced.

See also

- Natural Language Processing
- Knowledge Representation
- English Grammar
- Structured English
  - ClearTalk, another machine-readable knowledge representation language
  - Inform 7, a programming language with English syntax

References

1. Norbert E. Fuchs, Kaarel Kaljurand, Gerold Schneider (2006). "Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces" (PDF). *FLAIRS 2006*.
2. Norbert E. Fuchs, Kaarel Kaljurand, Tobias Kuhn (2010). "Discourse Representation Structures for ACE 6.6" (PDF). *Technical Report ifi-2010.0010,*

Department of Informatics, University of Zurich.

3. Tobias Kuhn (2007). "AceRules: Executing Rules in Controlled Natural Language" (PDF). *First International Conference on Web Reasoning and Rule Systems (RR 2007)*.
4. Kaarel Kaljurand, Norbert E. Fuchs (2007). "Verbalizing OWL in Attempto Controlled English" (PDF). *OWL: Experiences and Directions (OWLED 2007)*.

External links

- Project Attempto (http://attempto.ifi.uzh.ch)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Attempto_Controlled_English&oldid=699461549"

Categories: Controlled English | Knowledge representation | Controlled natural languages | Natural language processing | Natural language parsing