ACE can be described by itself. ⋆

Norbert E. Fuchs, Kaarel Kaljurand, Tobias Kuhn

Department of Informatics & Institute of Computational Linguistics
University of Zurich, Switzerland
{fuchs,kalju,tkuhn}@ifi.uzh.ch
http://attempto.ifi.uzh.ch

Abstract. Attempto Controlled English (ACE) [1] is a mature and expressive knowledge representation language. To test the limits of ACE's expressivity we describe ACE and its associated tools in ACE itself. While this abstract and the reference section use unconstrained English, the rest of the article is expressed in ACE and can be submitted to the Attempto Parsing Engine (APE). The plain text version of the content of this paper and the lexicon needed to parse it can be found online[1].

1 Who needs a controlled-natural-language?

There are some users who want to use a formal method and who understand no formal language. They prefer to use a formal method in their own natural language. If a formal method is expressed-in a natural language then every person who speaks the natural language can use the formal method.

Every controlled-natural-language is a subset of a natural language. If a controlled-natural-language can be translated-into a logical language then the controlled-natural-language is a formal language that can be easily understood by every user who speaks the natural language that underlies the controlled-natural-language. Every developer of a controlled-natural-language believes that it can combine the formality of a formal language and the familiarity of a natural language.

PENG and Common-Logic-Controlled-English and CPL and Rabbit and ACE are five important controlled-natural-languages. ACE and five tools that use ACE are presented here.

2 What is ACE?

Attempto-Controlled-English is a controlled-natural-language that is a subset of English and that is developed by the Attempto-team that is a research-group of the University-of-Zurich. "ACE" is an abbreviation of "Attempto-Controlled-English".

⋆ We would like to thank Michael Hess cordially for supporting the Attempto group over all the years and wish him all the best for the future.

[1] http://attempto.ifi.uzh.ch/site/pubs/festschrift_mh/

2.1 Which words are supported by ACE?

The vocabulary of ACE consists-of the function-words that are built-in and the content-words that are defined by a lexicon.

ACE uses more than 80 function-words. Three typical function-words are "every" and "or" and "is".

Every content-word is an adverb or is an adjective or is a proper-name or is a noun or is a verb or is a preposition. ACE uses a lexicon that contains more than 100000 content-words. Every user of ACE can use a lexicon that contains some additional content-words.

2.2 What is the syntax of ACE?

ACE is a rich language that is described by less than 20 construction-rules and less than 20 interpretation-rules.

The construction-rules describe the syntax of ACE. They constrain the form of all admissible sentences of ACE. The interpretation-rules define the meaning of ACE.

Every admissible sentence of ACE is a declarative sentence or is a question or is a command.

Every declarative sentence ends-with a period, and is a simple sentence or is a composite sentence. Every simple sentence consists-of a noun-phrase and a verb-phrase or consists-of "there is/are" and a noun-phrase. "John works." is an example of a simple sentence. Every composite sentence is recursively formed by at least one constructor. Every constructor is a coordination or is a subordination or is a quantification. "If John works then Mary does not wait." is an example of a composite sentence.

Every question is a yes-no-question or is a wh-question and every question ends-with a question-mark. "Does John wait?" is an example of a yes-no-question and "Who waits?" is an example of a wh-question.

Every command starts-with an addressee and ends-with an exclamation-mark. "John, wait!" is an example of a command.

Every ACE-text consists-of at least one sentence every noun-phrase of which can be referred-to by an anaphoric reference. Every anaphoric reference is a definite noun-phrase or is a pronoun or is a variable or is a proper-name. Every anaphoric reference refers-to a noun-phrase that precedes the anaphoric reference.

2.3 ACE is not ambiguous.

English is ambiguous. ACE is not ambiguous. The construction-rules of ACE and the interpretation-rules of ACE eliminate all ambiguities of every admissible sentence of ACE.

For every ACE-text there is exactly one logical formula that represents the ACE-text and that defines the logical meaning of ACE.

2.4 ACE is understood by its own users.

There is an experiment that tests the understandability of ACE and a formal language that is comparable-to ACE.

The result of the experiment shows that ACE is more understandable than the formal language and that the learning-time of ACE is shorter than the learning-time of the formal language.

3 Which tools use ACE?

There are some tools that use Attempto-Controlled-English. Every tool that uses ACE and that is developed by the Attempto-team is licensed-under LGPL. LGPL is an open-source-license.

Five tools are presented here.

3.1 APE is a parser.

APE is a parser that translates every ACE-text into a formula of first-order-logic. "APE" is an abbreviation of "Attempto-Parsing-Engine".

APE can check the syntax of every ACE-text. If an ACE-text contains a syntactical error then APE generates an error-message that points-to the location of the error and that describes the possible cause of the error.

If the syntax of an ACE-text is correct then APE translates the ACE-text into a discourse-representation-structure. Every discourse-representation-structure is equivalent-to a formula of first-order-logic.

For every ACE-text APE optionally generates a paraphrase of the ACE-text and optionally translates the ACE-text into OWL and SWRL and TPTP.

3.2 RACE is a reasoner.

Every ACE-text establishes a logical theory and every author of an ACE-text may be interested-in its logical consequences.

RACE is a tool that performs some logical deductions on ACE. RACE can show that an ACE-text is consistent or is inconsistent and RACE can show that an ACE-text is a logical consequence of an ACE-text and RACE can extract the answer of an ACE-question from an ACE-text.

Every input of RACE is expressed-in ACE. Every output of RACE is expressed-in ACE or is expressed-in English. RACE hides every technical detail from its own users.

RACE is supported by some auxiliary axioms that are expressed-in Prolog and that represent some domain-independent knowledge.

3.3 AceRules is a rule-engine.

AceRules is a forward-chaining rule-engine whose input is ACE and whose output is ACE. AceRules supports three semantics. Every input of AceRules is expressed-in ACE and every output of AceRules is expressed-in ACE. AceRules supports negation-as-failure and supports strong-negation.

3.4 AceWiki is a semantic-wiki.

AceWiki is a semantic-wiki that uses ACE. Every article of AceWiki is written-in ACE and is translated-into OWL. AceWiki uses Pellet that is a reasoner. AceWiki ensures the consistency of its own knowledge-base. If a user asks a question in ACE then AceWiki answers the question.

AceWiki uses a grammar that describes a subset of ACE. AceWiki contains a predictive editor that processes the grammar. If a user wants to extend the content of AceWiki then the predictive editor can help the user.

There are some experiments that test the usability of AceWiki. The result of the experiments shows that every untrained user can use AceWiki.

3.5 ACE_View is an ontology-editor.

ACE_View is an ontology-editor that supports ACE and that supports OWL and that supports SWRL.

An example of an OWL-axiom is "$town \sqsubseteq \exists part^{-}.country$". The example contains at least three symbols that confuse some users. The example is expressed-in ACE as "Every town is a part of a country.". If the example is expressed-in ACE then the users are not confused by the example.

Every reasoner that ACE_View embeds understands OWL or understands SWRL. If a user who uses ACE_View and who understands ACE does not understand OWL and does not understand SWRL then the user can develop an ontology that is expressed-in OWL or that is expressed-in SWRL.

4 The Attempto-team congratulates Michael.

Michael, have a happy birthday!

References

1. Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Attempto Controlled English for Knowledge Representation. In Cristina Baroglio, Piero A. Bonatti, Jan Małuszyński, Massimo Marchiori, Axel Polleres, and Sebastian Schaffert, editors, *Reasoning Web, 4th International Summer School 2008, Venice, Italy, September 7–11, 2008, Tutorial Lectures*, number 5224 in Lecture Notes in Computer Science, pages 104–124. Springer, 2008.