Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces

Norbert E. Fuchs and **Kaarel Kaljurand** and **Gerold Schneider**
Department of Informatics & Institute of Computational Linguistics
University of Zurich
Email: {fuchs,kalju,gschneid}@ifi.unizh.ch

Abstract

We present Attempto Controlled English — a user-friendly first-order logic language with a rich English syntax — and its associated tools, and demonstrate how they meet the challenges of knowledge representation, reasoning, interoperability and user interfaces created by large software projects like the semantic web.

Introduction

Large software projects, for instance the semantic web, create enormous challenges for the representation of knowledge, for reasoning, for interoperability and for user interfaces.

Languages like first-order logic, UML and OWL are generally considered to meet the challenges of knowledge representation, reasoning and interoperability. However, these languages are hard to understand by the non-initiated, and thus fail the challenge of providing generally acceptable user interfaces.

Concerning user interfaces, natural language excels as the prototypical means of human communication. Natural language is easy to use and to understand, and does not need an extra learning effort. Furthermore, natural language is highly expressive, and can be used in any application domain. Some researchers even consider natural language “the ultimate knowledge representation language” (Sowa 2000). On the other hand, natural language has traditionally been dismissed as too ambiguous and too resource-consuming to be suitable as a knowledge representation language. We believe that this rebuttal needs to be revisited, and that Sowa’s claim can be substantiated.

The past decade has brought spectacular progress for natural language processing (NLP). Thanks to the integration of statistics, both precision and recall in NLP have reached new levels (Collins 1999; Bod 2001). On the other hand, a sobering realisation is being made: precision and recall values of 80 to 95 % seem to be the ceiling for current statistical NLP approaches. To achieve reliable knowledge bases, human intervention would be required, a scenario that is not practical in general.

Fortunately, there has also been great progress in another line of NLP research, namely controlled natural languages (Controlled Natural Languages 2006). A controlled natural language is a subset of the respective natural language that is specifically designed to serve as a documentation, specification or knowledge representation language. The ambiguity and vagueness of full natural language can be avoided and efficient processing is possible.

In the last years, we have developed the specification and knowledge representation language Attempto Controlled English (ACE). For details check the Attempto website (Attempto project 2006). The current version 4 of ACE offers language constructs like countable and mass nouns, collective and distributive plurals, generalised quantifiers, indefinite pronouns, phrasal and prepositional verbs, noun phrase/verb phrase/sentence negation, and anaphoric references to noun phrases through proper names, definite noun phrases, pronouns, and variables.

Every ACE sentence conforms to standard English grammar and has an unambiguous meaning — even if the same sentence may appear ambiguous in unconstrained English.

Though appearing completely natural, ACE is a formal language defined by an abstract grammar and by a concrete grammar implemented by the Attempto Parsing Engine (APE). APE translates an ACE text into a first-order representation which allows us to assign ACE a formal semantics. In brief, ACE is a logic language with an English syntax.

Controlled natural languages have been developed by other groups for various purposes. For an excellent overview consult the Controlled Natural Languages Homepage (Controlled Natural Languages 2006). Closest to our work are Boeing’s Computer Processable Language (Clark *et al.* 2005), Schwitter’s Processable English (PENG 2006), and Sowa’s Common Logic Controlled English (Sowa 2004).

In the following sections we will show how ACE meets the challenges of knowledge representation, reasoning, interoperability, and user interfaces. Then we conclude and point to future research.

Knowledge Representation in ACE

Here we present only a brief overview of the language ACE. For a comprehensive description see (Fuchs *et al.* 2005a).

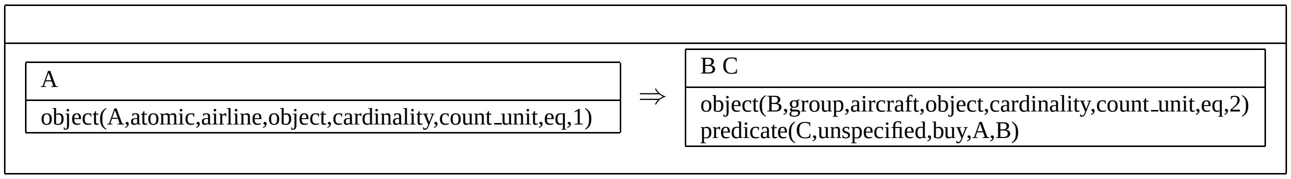
A
object(A,atomic,airline,object,cardinality,count_unit,eq,1)
⇒
B C
object(B,group,aircraft,object,cardinality,count_unit,eq,2)
predicate(C,unspecified,buy,A,B)

Figure 1: DRS corresponding to the ACE sentence “Every airline buys 2 aircraft.”

ACE Syntax An ACE text consists of anaphorically interrelated simple and composite sentences.

Simple sentences contain a verb that can be intransitive, transitive or ditransitive. Furthermore, there is the copula ‘be’. Verbs can be complemented by noun phrases and prepositional phrases (acting as subjects, direct objects and indirect objects), and modified by optional adverbs and prepositional phrases. Noun phrases contain a noun that must be preceded by a determiner, and that can be modified by adjectives, possessive pronouns, Saxon genitives, of-constructs, appositions (variables or strings) and relative clauses. Phrases of the same type can usually be coordinated.

Composite sentences are recursively built from simpler sentences by co-ordination (and, or, ...), quantification (every, all, for every, for all, there is, ...), negation (no, does/is not, it is not the case that, ...), and subordination (if-then).

Though the ACE syntax is simple, ACE sentences can get quite complex, for example

(1) Every big and red button ”Go!” of an upper panel that is illuminated and that is not inactivated controls the aircraft on the runway.

ACE also supports interrogative sentences, for instance

(2a) Is the button inactivated?
(2b) What does the button control?
(2c) Where does the button control the aircraft?

The ACE lexicon contains a fixed set of predefined function words, and a large set of content words (nouns, verbs, adjectives, adverbs) that users can extend. Each content word can be simple or compound. Verbs are restricted to simple present tense, third person singular and plural, active voice, and indicative mood.

ACE Semantics The semantics of an ACE text is defined by mapping it to Discourse Representation Structures (DRS), i.e. to first-order logic.

We have extended the standard language of DRSs (Kamp & Reyle 1993; Blackburn & Bos 1999) to express plurality in first-order logic and to be able to effectively formulate reasoning axioms (Fuchs *et al.* 2005b). Our DRS representation strikes a balance between the intuitively acceptable meaning of ACE sentences and computational tractability. The sentence

(3) Every airline buys 2 aircraft.

is translated into the DRS in figure 1. Logical atoms are formed from a small set of predefined predicates like `predicate/5`, that have as arguments reified predicates derived from content words. Reification allows us to quantify over the arguments of the predefined predicates, thus expressing general aspects of relations in first-order axioms that otherwise would require higher-order logic (Hobbs 1985). In order to deal with plural constructions (Schwertel 2004), information about the quantity of objects, and a distinction between countable and mass nouns are introduced into the DRS. Refer to (Fuchs *et al.* 2005b) for a comprehensive description of the extended DRS language.

Synonymy is an important means to render texts more natural. ACE provides synonymy both on the syntactical as on the lexical level. On the syntactical level, different ACE texts can be mapped to the same DRS, i.e. have the same semantics. For instance, sentence (3) can be replaced by the semantically equivalent sentence

(3’) If there is an airline then the airline buys 2 aircraft.

With regard to lexical synonymy, each content word can have any number of synonyms called aliases. During parsing, aliases are replaced by the main word.

Handling Ambiguity As a formal language, ACE is not ambiguous. To achieve this we employ three means.

First, some ambiguous constructs are not part of ACE; unambiguous alternatives are available in their place.

Second, all remaining ambiguous constructs are interpreted deterministically using a small set of interpretation rules. Syntactically marked ACE constructs exist to express alternative readings. This syntax-oriented approach renders disambiguation intelligible, reproducible and fast.

Third, the ACE parser generates a paraphrase in a subset of ACE called Core ACE that the users can accept or reject. If the users do not accept the paraphrase then they have to reformulate the input. Contrast this with approaches in which the parser generates all possible readings and then lets the user choose the intended one.

We now consider some examples of interpretation rules. Please note that the braces {} used in the examples are not part of ACE, and are only introduced to emphasise the interpretations.

- Quantifier scoping: The textual occurrence of a quantifier opens its scope that extends to the end of the sentence, respectively conjoined sentence.

  To express the two readings of the English sentence ‘Every airline owns an aircraft.’ one writes in ACE

  (4) {Every airline owns {an aircraft}}.
  (5) {There is an aircraft that {every airline owns}}.

- Sentential negation scoping: The scope of sentential negations does not include co-ordinated sentences. A wider scope can be forced by repeating the subordinating conjunction.

(6) {It is not the case that an aircraft waits} and a runway is empty.
(7) {It is not the case that an aircraft waits and that a runway is empty}.

- Plural interpretations: Plural noun phrases have collective reading. Distributive reading is expressed by the 'each of' construct.

(8) A technician services 2 aircraft.
(9) A technician services each of 2 aircraft.

- Lexical ambiguity: To resolve the lexical ambiguity between intransitive verbs (wait) plus prepositional phrase (on a runway) and prepositional transitive verbs (wait on somebody), ACE expects that the preposition of a prepositional verb is hyphenated to the verb.

(10) An aircraft waits {on a runway}.
(11) A stewardess {waits-on} a passenger.

For a complete listing of the ACE interpretation rules, refer to (Fuchs *et al.* 2005a).

Resolving Anaphoric References Anaphoric references are an important mechanism to achieve a coherent and naturally sounding text. Efficient resolution of anaphoric references contributes to a great degree to the overall efficiency of processing (controlled) natural language.

ACE provides proper names, pronouns, variables and definite noun phrases as anaphoric references to noun phrases.

Proper names always denote the same object and thus serve as their own anaphoric references. In all other cases anaphoric references refer to indefinite noun phrases, and their resolution is governed by accessibility, recency, specificity, and reflexivity. This renders resolution intelligible, reproducible, and furthermore highly efficient.

With a sole exception, classical DRS accessibility constraints apply. This means that a noun phrase is not accessible if it occurs in a negated or in a universally quantified context. A noun phrase introduced in the if-part of an if-then sentence is accessible in the then-part.

(12) A pilot does not have a valid licence. *It is expired.
(13) If an aircraft is on the runway then the aircraft has a running engine. *It does not run.

(NB. Sentences prefixed with * are not accepted by the ACE parser.)

Following DRS accessibility constraints, noun phrases introduced in a disjunction are not accessible outside of the disjunction. However, contrary to standard DRS accessibility constraints, a noun phrase introduced in a disjunct is accessible in subsequent disjuncts.

(14) An engine runs or it does not run. *It is serviced.

If the anaphor is a non-reflexive personal pronoun (he, him, ...), or a non-reflexive possessive pronoun (his, ...), or a relative pronoun 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.

(15) The aircraft of the pilot is serviced. Its engine runs. He is satisfied.
(16) An officer checks the licence of a pilot who is new. (= the pilot is new)
(17) An officer checks the licence of a pilot which is new. (= the licence is new)

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.

(18) A pilot looks-at his own licence and reassures himself.

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. Specificity includes all modifications of a noun, i.e. adjectives, Saxon genitives, of-relations, appositions and relative clauses.

(19) There is an aircraft that has a normal engine. There is an aircraft that has a special engine. There is an aircraft that has 2 engines. The pilot chooses the aircraft that has a normal engine. (= the pilot chooses the first aircraft)

If a definite noun phrase cannot be resolved then it is interpreted as an indefinite noun phrase introducing a new object.

If the anaphor is a variable then it refers to the noun phrase that introduces the variable.

(20) There is an aircraft X that has a normal engine. There is an aircraft Y that has a special engine. The pilot chooses X.

Core ACE Removing all synonymous syntactic constructs from ACE, we arrive at a subset of ACE called Core ACE. Core ACE is semantically equivalent to full ACE but lacks its syntactic variants.

Core ACE uses full sentences instead of relative phrases, allows only sentence negation, replaces universal quantification by if-then constructs, substitutes of-constructs for Saxon genitives, knows only definite noun phrase anaphors, and fixes the word order. Note that Core ACE does currently not cover distributive plural noun phrases and wh-interrogative sentences.

Core ACE can be used for several purposes, the main one being to paraphrase ACE texts. Such paraphrases fulfil the following requirements

- the paraphrase is semantically equivalent to the original,
- the mapping is deterministic,
- the paraphrase needs fewer interpretation rules to be understood.

E.g. the paraphrase of example (1) is

(21) If there is a big and red button ”Go!” of an upper panel and the upper panel is illuminated and it is not the case that the upper panel is inactivated then the big and red button ”Go!” of the upper panel controls an aircraft on a runway.

As a downside, some ACE texts (those that fall into the Core ACE subset) are not paraphrased into a different form. Also, due to the syntactic restrictions of Core ACE, paraphrases can sound less natural than the original ACE text.

The paraphrase of an ACE text is generated by translating the DRS derived from the text back into Core ACE (Fuchs, Kaljurand, & Schneider 2005). Parsing the paraphrase gives the same DRS and consequently the same paraphrase.

Implementation The Attempto Parsing Engine (APE) translates an ACE text into a DRS. APE is implemented in Prolog as a Definite Clause Grammar (DCG) using feature structures. APE relies on a lexicon of function words and a full-form lexicon of about 100,000 content words derived from COMLEX (Wolff, Macleod, & Meyers 1998; Bünzli 2004). The resolution of anaphoric references is implemented as a separate module that accepts an unresolved DRS with additional conditions for anaphors and potential antecedents and produces a resolved DRS as the final output.

APE is publicly available via a remote procedure call, implemented as a REST webservice. The service translates an ACE text into a DRS, and optionally provides information about the tokenisation, syntax, paraphrase, and classical first-order logic representations of the input text. As an example of using the webservice, we provide a webclient. The current version of the webclient allows to input an ACE text, supplement APE's internal lexicon with a user lexicon, and control the display of the output.

ACE for Knowledge Representation As a first-order logic language, ACE certainly fulfils the prerequisites of a knowledge representation language, but only its application to a range of problems can determine its usefulness and practicality. As the saying goes, the proof of the pudding is in the eating.

ACE was originally developed as a specification language, and was used, for instance, to specify an automated teller machine, Kemmerer's library data base (Schwitter 1998), data base integrity constraints (Fuchs, Schwertel, & Torge 2000), and Kowalski's subway regulations (Fuchs, Schwertel, & Schwitter 1999). Currently, ACE is used to express an ontology of proteins and their interactions (Kuhn 2006). In 2004, ACE was adopted as the controlled language for the EU Network of Excellence REWERSE (Reasoning on the Web with Rules and Semantics) (REWERSE 2006).

Since ACE has been successfully used by several researchers for a variety of problems, we confidently claim that it has proved to meet the challenge of a knowledge representation language.

Reasoning in ACE

RACE The Attempto Reasoner (RACE) supports automatic reasoning in ACE. Currently, RACE proves that one ACE text is the logical consequence of another one, and gives a justification for the proof in ACE. If there is more than one proof then RACE will find all of them. Variations of the basic proof procedure permit query answering and consistency checking.

Implementation The functionality of RACE is defined by a set of requirements (Fuchs & Schwertel 2003) that reflect our philosophy to make formal methods available to people who are not familiar with them. As the basis for RACE we took theorem provers and model generators available off-the-shelf, prototypically implemented RACE with each of them, and evaluated the prototypes with respect to the requirements. The best results were achieved with Otter and Satchmo. The current Prolog implementation of RACE is based on the model generator Satchmo (Fuchs & Schwertel 2003). Thus, RACE performs not only theorem proving but also model generation.

Satchmo (Manthey & Bry 1988) is implemented as a small, efficient Prolog program. Adapting it to fulfil RACE's requirements, and still preserve most of its efficiency, turned out to be no easy task. To compensate for the loss of efficiency caused by the additional functionality of RACE, we introduced optimisations like clause compaction, and informed search.

ACE axioms and ACE theorems are first translated into DRSs and then into first-order clauses in which the actual reasoning is performed. Reasoning is supported by auxiliary first-order axioms for the lattice-theoretic implementation of plurals, for the processing of equality and for other purposes. Prolog predicates are used for the operations on natural numbers. The results of a proof are reported using the original ACE axioms and theorems. In the spirit of the Attempto project, intermediate results are hidden from the users. Also, users need not set parameters to control proofs.

Examples Here is a simple example of using RACE for consistency checking. Given the ACE text

> Every airline that buys a standard aircraft gets a discount. A British airline buys a standard aircraft. A French airline buys a standard aircraft. There is no airline that gets a discount.

RACE finds two inconsistent subsets

> Every airline that buys a standard aircraft gets a discount. A French airline buys a standard aircraft. There is no airline that gets a discount.

> Every airline that buys a standard aircraft gets a discount. A British airline buys a standard aircraft. There is no airline that gets a discount.

Here is another simple example for theorem proving. Given the ACE axioms

> Every airline that buys an aircraft gets a discount. Each of 6 Swiss airlines buys an aircraft.

and the ACE theorem

> An airline gets a discount.

RACE proves that the sentence

> An airline gets a discount.

can be deduced from the sentences

> Every airline that buys an aircraft gets a discount. Each of 6 Swiss airlines buys an aircraft.

using the auxiliary axioms

(Ax. 9): Definition of proper_part_of.
(Ax. 10-1): Every group consists of atomic parts.
(Ax. 22-1): Number Axiom.

Notice that three auxiliary axioms (and also hidden Prolog predicates for natural numbers) are needed to prove the theorem.

Until now, RACE has only been applied to a few smaller problems like Schubert's steamroller, and Lewis Carroll puzzles. A realistic application is planned within ongoing work on protein ontologies. For the time being, we can only state that RACE offers a promising approach to reasoning in ACE, but that convincing results are still missing.

Interoperability Support

Interoperability among applications, for instance those distributed over the WWW, is a complex technical and organisational problem. To support interoperability we provide translations of ACE into and from other languages.

All of these translations rely on DRSs as interlingua. While the Attempto Parsing Engine (APE) translates ACE into DRSs, the reverse translation of DRSs into Core ACE is performed by the complementary tool DRACE.

First-Order Languages A DRS can be further translated into formal languages equivalent to the language of first-order logic. For example, translations into the standard and the clausal form of first-order logic are used within RACE. The reverse translation of first-order logic expressions into DRSs is already defined (Fuchs, Kaljurand, & Schneider 2005), and is about to be implemented.

Subsets of First-Order Languages Transformations into languages equivalent to subsets of first-order logic — e.g. the languages PQL (Bernstein, Kaufmann, & Fuchs 2005), PRQ (Fuchs, Schwertel, & Torge 2000), FLUX (Dawelbait 2004) that were used in practical applications of ACE — usually ignore some information of the DRS. Transformation of a DRS into semantic web languages also fall into this category. Concretely, in a prototypical implementation we already translate ACE into a subset of OWL. In another project, David Z. Hirtle at the University of New Brunswick translates ACE into RuleML. There are not yet any transformations of these languages into DRSs; we plan, however, a translation of OWL DL into DRSs.

On the basis of these results, we believe that ACE promises to be a valuable support for interoperability.

ACE as User Interface

Natural language as the prototypical means of human communication is an optimal means to communicate with computer systems. With some reservations concerning expressivity, this is also true for controlled natural languages like ACE. Again, the proof of the pudding is in the eating, i.e. in concrete applications — of which there are several.

ACE served as natural language interface for the model generator EP Tableaux (Fuchs, Schwertel, & Torge 2000), for a FLUX agent (Dawelbait 2004), and for MIT's Process Handbook (Bernstein, Kaufmann, & Fuchs 2005).

In (Kuhn 2006) ACE was used instead of OWL to describe an ontology of protein interactions.

Furthermore, ACE is being investigated for policy and business rules within the EU Network of Excellence REWERSE (REWERSE 2006). The feedback of the users was in general positive, and resulted in constructive proposals for extensions of ACE.

In a controlled experiment (Bernstein *et al.* 2005), CS students preferred ACE to SQL as database query language. The students designed ACE queries resulting in a very good retrieval performance (100% precision and 90% recall). This tallies with our experience that ACE can be learned in 2–3 days with little previous knowledge.

On the basis of this we claim that ACE provides an excellent basis for user interfaces.

Conclusions

We have presented Attempto Controlled English (ACE) and associated tools, and assessed how far they meet the challenges of knowledge representation, reasoning, interoperability and user interfaces. The results of this assessment are generally positive, or at least encouraging, and show that controlled languages form a promising bridge between natural language processing and knowledge representation.

We now discuss further work.

Knowledge Representation Currently, declarative and interrogative ACE sentences are allowed. Following requests from ACE users, we consider restricted forms of modality, negation as failure, and support for prioritised rules.

Reasoning Currently, RACE performs only deduction. Further forms of reasoning, for instance hypothetical reasoning ('What happens if ...?'), abductive reasoning ('Under which conditions ...?') and temporal reasoning are planned.

Also, ACE will be applied to rule-based programming, and to Grosof's Courteous Logic Programming (Grosof 1999) for which we have already two interpreters accepting rules expressed in Prolog (Dörflinger 2005).

Interoperability Once we will have completed the translation of first-order expressions into DRSs, we will have a complete bidirectional mapping between ACE and first-order logic, which can then complement DRSs as interlingua. In addition, we are currently investigating a full bidirectional mapping from a subset of ACE into OWL DL.

User Interfaces Many users of ACE see its potential as interface language and have suggested relevant extensions and enhancements. One of those proposals concerns extending ACE by imperatives. We have already developed a proposal that is based on minimal extensions of the language, and on processing imperative sentences as declarative ones. Extending ACE by imperatives would allow us to use it in a dialogue system, for instance to control interactively the FLUX agent described in (Dawelbait 2004).

A mapping between ACE and OWL would enable ontology building entirely on the level of ACE, and could replace the complex graphical front-ends to OWL.

Acknowledgements

The authors would like to thank Stefan Höfler, Fabio Rinaldi, Uta Schwertel and Rolf Schwitter for their contributions to the Attempto project. Further thanks go to the anonymous reviewers for their constructive comments.

References

Attempto project. 2006. Attempto project website. `http://www.ifi.unizh.ch/attempto`.

Bernstein, A.; Kaufmann, E.; Göhring, A.; and Kiefer, C. 2005. Querying Ontologies: A Controlled English Interface for End-users. In *4th International Semantic Web Conference*.

Bernstein, A.; Kaufmann, E.; and Fuchs, N. E. 2005. Talking to the Semantic Web — A Controlled English Query Interface for Ontologies. *AIS SIGSEMIS Bulletin* 2(1).

Blackburn, P., and Bos, J. 1999. *Working with Discourse Representation Theory. An advanced Course in Computational Semantics*.

Bod, R. 2001. What is the Minimal Set of Fragments that Achieves Maximal Parse Accuracy? In *Proceedings of ACL-2001*.

Bünzli, A. 2004. AceLex — Lexikon für Ace. MA thesis. Institute of Computational Linguistics, University of Zurich.

Clark, P.; Harrison, P.; Jenkins, T.; Thompson, J.; and Wojcik, R. H. 2005. Acquiring and Using World Knowledge Using a Restricted Subset of English. In *FLAIRS 2005*.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. Dissertation, University of Pennsylvania, Philadelphia, PA.

Controlled Natural Languages. 2006. Controlled Natural Languages. `http://www.ics.mq.edu.au/~rolfs/controlled-natural-languages`.

Dawelbait, G. 2004. Attempto Controlled English (ACE) as a Communication Language For a FLUX Agent. MSc thesis. Department of Computer Science, Dresden University of Technology.

Dörflinger, M. 2005. Interpreting Courteous Logic Programs. Diploma Thesis. Department of Informatics, University of Zurich.

Fuchs, N. E., and Schwertel, U. 2003. Reasoning in Attempto Controlled English. In *Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2003)*, number 2901 in Lecture Notes in Computer Science. Springer.

Fuchs, N. E.; Höfler, S.; Kaljurand, K.; Rinaldi, F.; Schneider, G.; and Schwertel, U. 2005a. Attempto Controlled English (ACE), Language Manual, Version 4.0. Technical report, Department of Informatics, University of Zurich. Forthcoming.

Fuchs, N. E.; Höfler, S.; Kaljurand, K.; Schneider, G.; and Schwertel, U. 2005b. Extended Discourse Representation Structures in Attempto Controlled English. Technical Report ifi-2005.08, Department of Informatics, University of Zurich, Zurich, Switzerland.

Fuchs, N. E.; Kaljurand, K.; and Schneider, G. 2005. Deliverable I2-D5. Verbalising Formal Languages in Attempto Controlled English I. Technical report, REWERSE. `http://rewerse.net/deliverables.html`.

Fuchs, N. E.; Schwertel, U.; and Schwitter, R. 1999. Attempto Controlled English — Not Just Another Logic Specification Language. In Flener, P., ed., *Logic-Based Program Synthesis and Transformation*, number 1559 in Lecture Notes in Computer Science. Manchester, UK: Eighth International Workshop LOPSTR'98.

Fuchs, N. E.; Schwertel, U.; and Torge, S. 2000. A Natural Language Front-End to Model Generation. *Journal of Language and Computation* 1(2):199–214.

Grosof, B. N. 1999. Compiling Prioritized Default Rules into Ordinary Logic Progams. Technical Report RC 21472, IBM Research, IBM T.J. Watson Research Center.

Hobbs, J. R. 1985. Ontological promiscuity. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, 60–69. Morristown, NJ, USA: Association for Computational Linguistics.

Kamp, H., and Reyle, U. 1993. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers.

Kuhn, T. 2006. Expressing Ontological Knowledge of Protein Interactions in Attempto Controlled English. Diploma Thesis. Department of Informatics, University of Zurich.

Manthey, R., and Bry, F. 1988. SATCHMO: A Theorem Prover Implemented in Prolog. In *CADE 88, Ninth International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, 415–434. Argonne, Illinois: Springer.

PENG. 2006. PENG website. `http://www.ics.mq.edu.au/~rolfs/peng/`.

REWERSE. 2006. REWERSE website. `http://rewerse.net`.

Schwertel, U. 2004. *Plural Semantics for Natural Language Understanding – A Computational Proof-Theoretic Approach*. Ph.D. Dissertation, University of Zurich.

Schwitter, R. 1998. *Kontrolliertes Englisch für Anforderungsspezifikationen*. Ph.D. Dissertation, Department of Computer Science, University of Zurich.

Sowa, J. F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks Cole Publishing Co.

Sowa, J. F. 2004. Common Logic Controlled English. Technical report. Draft, 24 February 2004, `http://www.jfsowa.com/clce/specs.htm`.

Wolff, S. R.; Macleod, C.; and Meyers, A. 1998. COMLEX Word Classes Manual. Technical report, New York University.