ACE View — an ontology and rule editor based on Attempto Controlled English

Kaarel Kaljurand

Institute of Computational Linguistics, University of Zurich
kalju@ifi.uzh.ch

Abstract. We describe the architecture of a novel ontology and rule editor ACE View. The goal of ACE View is to simplify viewing and editing expressive and syntactically complex OWL/SWRL knowledge bases by making most of the interaction with the knowledge base happen via Attempto Controlled English (ACE). This makes ACE View radically different from current OWL/SWRL editors which are based on formal logic syntaxes and general purpose graphical user interface widgets. ACE View integrates two mappings, ACE→OWL/SWRL and OWL→ACE, and is implemented as a plug-in for Protégé 4.

1 Introduction

We describe the architecture of a novel ontology and rule editor ACE View. The goal of ACE View is to simplify the exploration and editing of expressive and syntactically complex OWL 2 [12] ontologies and SWRL [7] rulesets by basing the user interface on Attempto Controlled English (ACE) [3]. This makes ACE View radically different from current OWL/SWRL editors which are based on formal logic syntaxes and general purpose graphical user interface widgets (trees, checkboxes, etc.), and which are often seen as too complicated and confusing for domain experts with no background in formal methods [5]. ACE View integrates two mappings, ACE→OWL/SWRL and OWL→ACE, and is implemented as a plug-in for Protégé 4[1].

The emerging OWL 2 specification describes several serialization syntaxes for OWL ontologies (RDF and XML based, functional-style, Manchester OWL Syntax). Some of these syntaxes are oriented towards machines and are thus inherently difficult to read and write for humans. Others have been designed for logicians and programmers, but lack the features that would bring OWL closer to domain experts who are often not well-trained in formal methods. E.g. [13,9] list the problems that users encounter when working with OWL. While some of the problems are purely semantic (e.g. caused by misunderstanding the open world reasoning and the unique name assumption) and would be encountered in any syntax, many problems are rooted in the nature of current OWL syntaxes. Furthermore, many knowledge bases require a rule component, often expressed in SWRL. The proposed SWRL syntax, however, is completely different from the

[1] http://protege.stanford.edu/

OWL syntaxes (mainly because it explicitly uses variables) even though there is an overlap of the semantics of OWL and SWRL. Query languages for OWL ontologies introduce yet another set of syntaxes.

The syntactic complexity can be hidden to some extent by front-end tools such as Protégé which use various user interface widgets to support viewing and editing knowledge bases. For example, the sub class hierarchy of named classes can be presented as a tree. Still, the relative richness of OWL and related languages means that for more complex expressions (negation, property restrictions, etc.), the user interface has to fall back to one of the standard syntaxes.

An alternative and less explored approach is to base ontology editing on the use of controlled natural language (CNL) [15]. Several studies have shown that controlled English can offer an improved usability over existing approaches for domain experts working with OWL statements [10,4,5]. However, ontology editors that offer CNL-based interaction as their main component are still in their infancy and their possible architecture has not been agreed upon.

This paper describes ACE View, an ontology and rule editor that offers the creation, viewing, editing and querying of the logical content of OWL 2 ontologies and SWRL rulesets in ACE. ACE View offers one unified syntax for OWL axioms, SWRL rules and DL queries — axioms and rules are expressed as English declarative sentences and queries as English interrogative sentences. ACE View is implemented as an extension to the popular ontology editor Protégé. This greatly simplifies the implementation of our approach as we can leverage the integrated OWL API [6], reasoners, rule and query support that Protégé provides and just concentrate on providing the controlled English front-end to these features. Also, we can easily fall back to the Protégé solutions for e.g. annotation editing, etc. that we do not intend to express in ACE. Being based on Protégé also simplifies the evaluation of our approach, e.g. one sensible way to evaluate ACE View is to let users complete an ontology engineering task and observe for how much of it they want to fall back to the standard Protégé approach.

This paper is structured in the following way. In section 2 we give a short overview of ACE and the mappings between ACE and OWL/SWRL, in section 3 we describe the main features of the ACE View editor, in section 4 we review the related work, and finally, in section 5 we draw conclusions and describe future work.

2 ACE ⇔ OWL/SWRL

ACE is a subset of English, such that each sentence in the chosen subset is interpreted unambiguously, relating the sentence to a unique logical form. The intention behind the design of ACE is to offer expressivity required in knowledge engineering tasks, but also remain a natural subset of English. The design minimizes what the users need to learn (assuming knowledge of English) to correctly compose ACE sentences and understand their meaning, e.g. to predict the paraphrases or the logical entailments of the sentences.

In order to make ACE interoperable with some of the existing Semantic Web languages, mappings have been developed to relate ACE to OWL, SWRL, and DL-Query (see a detailed description in [8]). For example, the mapping of ACE to OWL/SWRL translates the ACE text

Every employee that does not own a car owns a bike.
Every man that owns a car likes the car.
Which car does John own?

into a combination of OWL axiom, SWRL rule and DL-Query (an OWL class expression).

$employee \sqcap \neg (\exists\ own\ car) \sqsubseteq \exists\ own\ bike$
$man(?x) \wedge own(?x, ?y) \wedge car(?y) \rightarrow like(?x, ?y)$
$car \sqcap \exists\ own^{-}\ \{John\}$

The OWL→ACE mapping, on the other hand, allows us to verbalize existing OWL ontologies as ACE texts. This mapping is not just the reverse of the ACE→OWL mapping as it also covers OWL axiom and expression types that the ACE→OWL mapping does not generate. For example, the OWL axiom

```
PropertyDomain(ObjectProperty(write) Class(author))
```

is verbalized as "Everything that writes something is an author.".

The mappings between ACE and OWL/SWRL provide an alternative syntax for OWL and SWRL. This syntax is readable as standard English and provides linguistically motivated syntactic sugar. It also makes the difference between OWL, SWRL and DL-Query invisible. This syntax is mainly intended for structurally and semantically complex OWL/SWRL knowledge bases for which visual methods and traditional syntaxes fail to provide a user-friendly front-end.

3 ACE View

3.1 Introduction

The ACE View editor provides an alternative view to the ontology and its entities — a natural language rendering of the complete logical content of the ontology where for the natural language we use ACE. In this rendering, ACE sentences correspond to OWL axioms, ACE words to OWL entities, and most metrics are linguistic, e.g. number of sentences and content words in the ACE text.

The ACE View editor lets the user manage an ACE text. The ACE text is a set of ACE snippets where each snippet is a sequence of one or more ACE sentences. The sentences in the snippet can be anaphorically linked. Usually, however, a snippet contains just a single sentence. When a snippet is added to the text, it is automatically parsed and converted into OWL/SWRL. If the translation fails then the snippet is still accepted, it simply does not have any logical axioms attached and thus cannot participate in reasoning. In case the translation succeeds, the snippet is mapped to one or more OWL axioms and

SWRL rules which are also merged into the Protégé managed ontology. In case a snippet is deleted, its corresponding axioms (if present) are removed from the underlying ontology.

Alternatively, the ACE View user can switch to one of the standard Protégé views to perform an ontology editing task. In case an OWL axiom is added in the standard view, then it is automatically verbalized into at most one snippet and merged into the ACE text.[2] If the verbalization fails (e.g. the verbalizer does not support the *FunctionalProperty*-axiom with data properties) then an error message is stored and the axiom is preserved in the ACE text in Manchester OWL Syntax. In case an axiom is deleted, then its corresponding snippet is deleted as well. Note that the deletion operations can sometimes be quite complex because the deleted axiom might have been generated by several different snippets. Furthermore, each of those snippets might have generated also other axioms.

The ACE text (and thus the ontology) can be viewed and edited at several levels (word, snippet, vocabulary, text). Word level provides an access to OWL entities in the ontology and allows one to specify how the entities should appear in ACE sentences, i.e. what are the surface forms (e.g. singular and plural nouns and verbs) of the words that correspond to the entities. Entities can be further annotated using the standard Protégé views. Snippets can be categorized as asserted declarative snippets, asserted interrogative snippets (i.e. questions) and entailed (declarative) snippets. Asserted snippets are editable and provide access to their details (parsing results such as error messages or syntax trees/syntax aware layout, corresponding axioms/rules, ACE paraphrase). Questions provide additionally answers. Entailed snippets are not editable but can be explored to find out the reasons that cause the entailment. Vocabulary is a set of ACE content words. It can be sorted alphabetically or by frequency of usage. As content words correspond to OWL entities, standard Protégé views offer even more presentation options, e.g. the "back-bone hierarchy" of sub class and "part of" relations; separation of the vocabulary into classes, properties, individuals. The vocabulary level provides an quick access to the word level, each selected/searched word (entity) can be automatically shown in the word level, or its corresponding snippets in the text level. An ACE text is a set of ACE snippets. This set can be filtered, sorted, and searched. Reasoning can be performed on the whole text to find out about its (in)consistency. A new text can be generated by filling it with snippets that the asserted text entails.

3.2 Views

The ACE View editor comprises several views, most of which are editable. Some views show what has been asserted, some show what is entailed. The following screenshots show working with the "People and pets" ontology[3] which we verbalized automatically in ACE and edited slightly to make it conform to the best

[2] The default Protégé user interface currently only allows SWRL rules to be viewed, but not edited and created.

[3] `http://protege.cim3.net/file/pub/ontologies/people.pets/people+pets.owl`

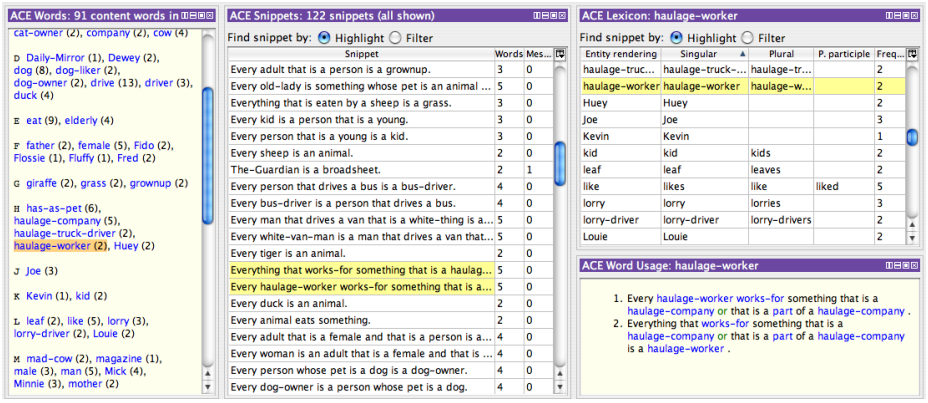

Fig. 1. One possible layout of the ACE View editor. Several views are shown: alphabetically ordered list of ACE content words (with frequency of usage in parentheses); ACE Snippets table, sortable by the length of the snippet, number of system messages, etc.; ACE Lexicon editor, highlighting the selected word and showing its morphological forms; ACE Word Usage view, showing all the snippets that contain the selected word.

practices of writing ACE (e.g. the construct 'has_part' was replaced by 'is a part of').

Vocabulary and wordform views (figure 1) In the "Lexicon view" and "Words view", the complete content word vocabulary of the ACE text is presented, sorted either alphabetically or by frequency of usage. Clicking on one of the words, "selects" this word signaling the other views (e.g. "Word Usage view") to display information about this word. In English, a content word can take several forms when used in a sentence. For example, the transitive verb 'border' (which maps to an object property 'border') can be used in three different forms.

> Every country that **borders** more than 2 countries that **border** a country that is **bordered** by a sea ...

The "Lexicon view" allows the user to edit such surface forms and make sure that they all correspond to the same OWL entity. When a new entity is generated in the standard Protégé views, the surface forms of its corresponding content word are automatically generated based on rules of English morphology. The user can override these forms if needed.

Views to asserted knowledge (figure 1) The "Snippets view" organizes all the asserted snippets in a table. With each snippet a set of its features are presented:

snippet length (in content words), the number of system error/warning messages, creation time, etc. The table rows can be highlighted and filtered based on the selected word, presenting only the snippets that contain the word. The "Snippet Editor" (figure 2) shows the logical and linguistic properties of the selected snippet and lets the user to edit the snippet. For sentences that fail to map to OWL/SWRL, error messages are provided. Error messages point to the location of the error and explain how to deal with the problem. Although the user is expected to enter sentences that can be mapped to OWL/SWRL, inputting sentences that are not compatible with the ACE→OWL/SWRL mapping or that are not even in ACE is tolerated. Such sentences, however, do not contribute to the text logically (e.g. they do not participate in reasoning). Such sentences can be modified at any time to make them comply with OWL/SWRL.

A paraphrase is one way for the user to check if his/her interpretation of the inserted text is accurate. ACE provides many forms of syntactic sugar, allowing to paraphrase *every*-sentences as *if-then* sentences, verb phrase negation as sentence negation, etc.

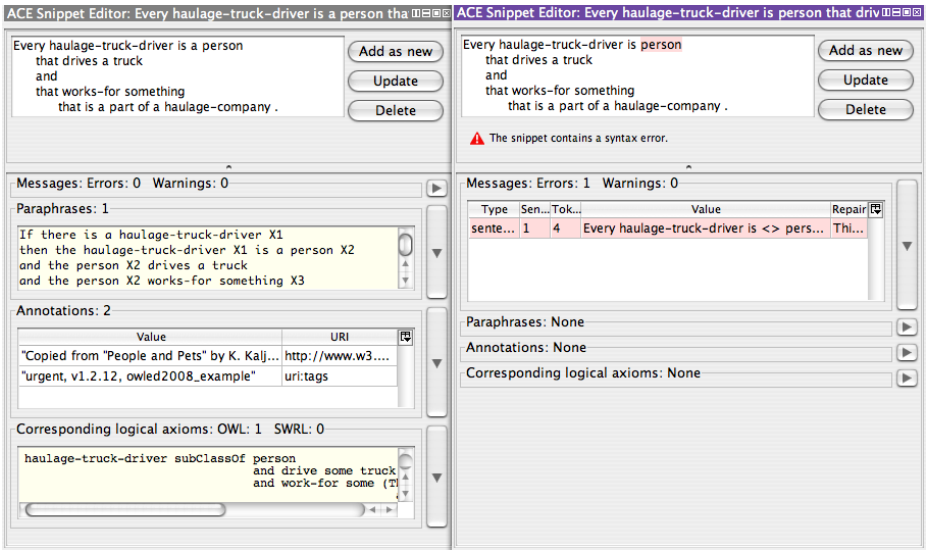

Fig. 2. ACE Snippet Editor. On the left: a successfully parsed sentence pretty-printed; feedback to the user in the form of a paraphrase and corresponding logical axioms; meta information in the form of OWL axiom annotations of the corresponding axioms. On the right: a sentence with a syntax error and an error message that points to the location of a missing determiner.

Views to entailed knowledge (figure 3) The "Q&A view" lists ACE questions and answers to them. These questions correspond to DL-Queries which are essentially (possibly complex) class expressions. The answers to a DL-Query are named individuals (members of the queried class) or named classes (named super and sub classes of the queried class). In ACE terms, the answers are ACE content words — proper names and common nouns. While the answers to DL-Queries are representation-wise identical in the ACE view and in the standard Protégé view, the construction of the query is potentially much simpler in the ACE view, as one has to construct a natural language question.

The "Entailments view" provides a list of ACE sentences that follow logically from the ACE text, i.e. these sentences correspond to the entailed axioms of the ontology. Such axioms are generated by the integrated reasoner on the event of classification. These axioms have a very simple structure, i.e. they are class assertions, property assertions and sub class axioms where the involved individuals, properties, and classes are always named. Complex class expressions including e.g. property restrictions are not part of the entailed axioms as provided by the reasoners currently integrated into Protégé. As the entailments are structurally simple, their natural language verbalization does not bring significant usability improvement over a traditional OWL syntax. Nevertheless, the presentation of all entailments as a single list of natural language sentences can provide a good and easily readable overview.

Protégé also supports entailment explanations. Such an explanation is a sequence of previously asserted axioms that motivate the entailment. The axioms in this sequence can be of any complexity and thus their natural language presentation can bring significant improvement in understanding the reason behind the entailment.

3.3 Benefits of using ACE View

As already discussed above, using ACE View brings several benefits to a domain expert building an ontology. In this section we list three more concrete beneficial features of ACE View.

Naming conventions and consistency By using English sentences as the main component of the user interface, ACE View puts a natural restriction on the orthography of OWL entity names and makes sure that the names are used consistently. For example, it would be quite hard for the user to mix the use of both singular nouns and plural nouns for class names. It would also be quite unnatural to use camel case in anything else than proper names (e.g. "easyJet"). A similar naming style is also promoted by the biomedical ontologies community [14].

Structural complexity of class expressions ACE View places a natural restriction on the complexity of OWL class expressions which by OWL syntax can be arbitrarily long lists and embed into each other to arbitrary depth. As

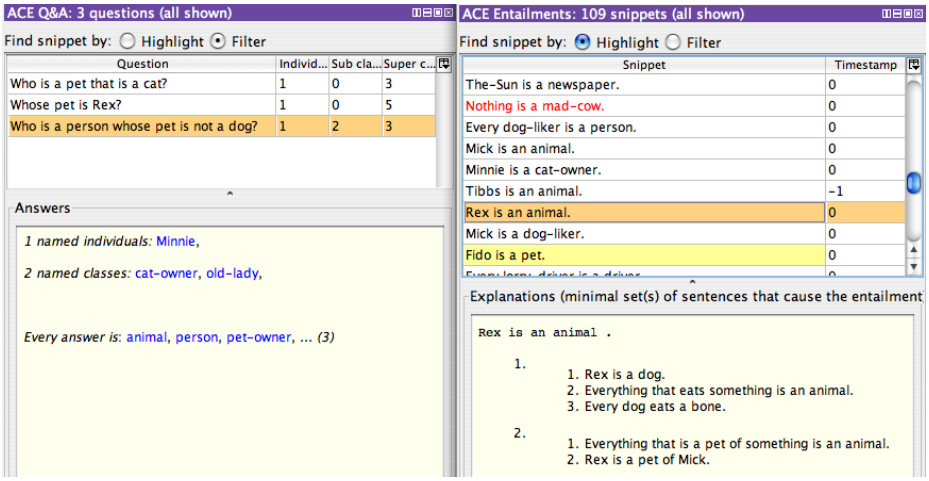

Fig. 3. The "Q&A view" lists all the entered questions with their automatically generated answers. The "Entailments view" lists the entailments that the ACE text makes. Clicking on an entailed snippet reveals the explanation — the asserted snippets that cause the entailment. Both the answers and the entailments/explanations are updated whenever the reasoner is run.

OWL classes are expressed by ACE noun phrases, users are more likely to create structurally simpler class expressions which in turn remain more human-readable in whatever syntax the are later displayed.

Fall-back to existing syntaxes/methods OWL 2 includes powerful shorthand axioms like *DisjointUnion*, motivated by common OWL usage patterns. ACE does not provide such short-hands and the corresponding ACE snippet will therefore unravel the complex construction via simpler ones. For example,

```
DisjointUnion(Class(person) Class(male) Class(female))
```

would be verbalized as

> Every person is a male or is a female. Everything that is a male or that is a female is a person. No male is a female.

While this is a valid approach that explains the notion of a covering union of pair-wise disjoint classes to a novice OWL user, more experienced OWL users may prefer a more concise representation that would give a better overview of what is being said. For that, ACE View allows the user to fall-back to a standard Protégé solution, e.g. a visual expression (*DisjointUnion* can be expressed as a pie chart) or Manchester OWL Syntax.

Also, many important statements in the ontology, e.g. about the scope and the purpose of the ontology, are non-logical and thus not handled by ACE View

via ACE sentences. As ACE View is implemented as a Protégé plug-in, it is very easy for the user to switch to standard Protégé annotation views for such meta-annotation tasks. Protégé also provides important editing support, such as refactoring (e.g. renaming entities), storage (ACE snippets are stored as axiom annotations and morphological surface forms as entity annotations), search, etc. ACE View will immediately profit from future developments in Protégé (e.g. collaborative editing).

3.4 Implementation and availability

The ACE parser has been implemented in SWI-Prolog and released under the LGPL open source license. The distribution also includes the translator from ACE to OWL/SWRL[4]. The ACE parser is also available as a REST-webservice[5].

The verbalization of OWL ontologies has been implemented in SWI-Prolog and is publicly available as a REST-webservice[6] that accepts ontologies in OWL 2 XML serialization as input and produces the an ACE text as output.

ACE View is implemented as a plug-in for Protégé 4 and relies heavily on the OWL API [6] that provides a connection to reasoners, DL-Query support, entailment explanation support, storage of OWL axioms and SWRL rules in the same knowledge base, etc. The main task of the ACE View plug-in, translating to and from OWL/SWRL, is performed by the two translator webservices. For the morphological generation of entity surface forms, we use the Lexicon Generation API[7] by Albert Gatt. ACE View plug-in is available in binary form[8]. We are currently working towards releasing it under an open source license.

4 Related work

AceWiki[9] is a "semantic wiki" [10,11] that uses ACE as its underlying formal language. Similarly to ACE View, it uses the ACE→OWL/SWRL mapping to enable reasoning over the wiki content. Semantic feedback is provided via question answering and consistency checking. An interesting feature of AceWiki is that editing of the wiki content is supported by a predictive editor, which in addition to auto-completing content words is also "syntactically aware" of the following context and can thus guide novice ACE users in forming syntactically correct ACE sentences.

The ROO tool [1] uses the Rabbit language [5] as its underlying controlled English. It is similar to ACE View as it has also been implemented as an extension to Protégé 4, although all of the default Protégé views have been stripped from the user interface. ROO allows entering Rabbit sentences which are then

[4] http://attempto.ifi.uzh.ch/site/downloads/
[5] http://attempto.ifi.uzh.ch/site/docs/ape_webservice.html
[6] http://attempto.ifi.uzh.ch/site/docs/owl_to_ace.html
[7] http://www.csd.abdn.ac.uk/~agatt/home/links.html
[8] http://attempto.ifi.uzh.ch/aceview/
[9] http://attempto.ifi.uzh.ch/acewiki/

translated into OWL automatically. The other direction of viewing existing OWL axioms as Rabbit sentences is not possible. Also, ROO does not provide any semantic feedback to the users. ROO was compared to ACE View in an evaluation involving 16 students who were beginners in conceptual modeling and OWL [2]. ROO was found to outperform ACE View in certain usability aspects. It must be noted though that the evaluation focused mainly on the syntactic aspects of using a CNL-based interface (e.g. how useful are the syntax error messages that the tool provides), and that some of these aspects have been improved in ACE View since the evaluation.

The controlled English CLoNE is being used in a GATE[10]-based ontology editor [4]. CLoNE, however, is quite a simple language (defined by 11 sentence patterns) and does not therefore provide the same syntactic and semantic expressivity as offered by ACE View.

A longer overview of work related to (C)NL-based ontology editors and query interfaces is provided by [16].

5 Conclusions and future work

ACE View introduces a novel paradigm to OWL/SWRL engineering. We assume that ontologies and rulesets are usually first expressed (in the minds of domain experts) in natural language, and thus working in ACE involves fewer conceptual problems. On the other hand, the combination of natural language based ontology editing and the standard form/formula-based editing offers more alternatives for the user and can result in an interesting synergy especially in the case of novice ontology engineers and domain experts working with semantically expressive and syntactically complex knowledge bases.

Our future work will focus on providing even more semantic feedback to the user, e.g. we would like to offer redundancy checking of the ontology, tracking changes in the entailments and answers (this can be seen as unit testing or regression testing known from software engineering), more choice for paraphrasing the asserted snippets, etc. To support the user syntactically, a predictive editor (e.g. the one used in AceWiki) could be integrated into ACE View.

Acknowledgment

This research has been funded by the European Commission and by the Swiss State Secretariat for Education and Research within the 6th Framework Programme project REWERSE number 506779 (cf. `http://rewerse.net`). The author is currently supported by the Swiss National Science Foundation (grant 100014-118396/1).

The author would like to thank the three anonymous reviewers of OWLED'08 for their valuable feedback. Also, the author would like to thank Norbert E. Fuchs, Tobias Kuhn, and Fabio Rinaldi for useful comments on the draft of this paper.

[10] `http://gate.ac.uk/`

References

1. Ronald Denaux, Ian Holt, Vania Dimitrova, Catherine Dolbear, and Anthony Cohn. Supporting the Construction of Conceptual Ontologies with the ROO Tool. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC)*, Washington, 1–2 April 2008.
2. Vania Dimitrova, Ronald Denaux, Glen Hart, Catherine Dolbear, Ian Holt, and Anthony Cohn. Involving Domain Experts in Authoring OWL Ontologies. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2008)*, Karlsruhe, Germany, 2008.
3. 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.
4. Adam Funk, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham, Brian Davis, and Siegfried Handschuh. CLOnE: Controlled Language for Ontology Editing. In *Proceedings of the Sixth International Semantic Web Conference (ISWC)*, Busan, Korea, 2007.
5. Glen Hart, Martina Johnson, and Catherine Dolbear. Rabbit: Developing a Controlled Natural Language for Authoring Ontologies. In *ESWC 2008*, 2008.
6. Matthew Horridge, Sean Bechhofer, and Olaf Noppens. Igniting the OWL 1.1 Touch Paper: The OWL API. In Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, editors, *3rd OWL Experiences and Directions Workshop (OWLED 2007)*, volume 258. CEUR Proceedings, 2007.
7. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004. Technical report, W3C, 2004. `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`.
8. Kaarel Kaljurand. *Attempto Controlled English as a Semantic Web Language*. PhD thesis, Faculty of Mathematics and Computer Science, University of Tartu, 2007.
9. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing Unsatisfiable Concepts in OWL Ontologies. In *ESWC 2006*, 2006.
10. Tobias Kuhn. AceWiki: A Natural and Expressive Semantic Wiki. In *Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*, 2008.
11. Tobias Kuhn. AceWiki: Collaborative Ontology Management in Controlled Natural Language. In *Proceedings of the 3rd Semantic Wiki Workshop*, volume 360. CEUR Proceedings, 2008.
12. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Working Draft 08 October 2008. Technical report, W3C, 2008. `http://www.w3.org/TR/2008/WD-owl2-syntax-20081008/`.
13. Alan L. Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In Enrico Motta, Nigel Shadbolt, Arthur Stutt, and Nicholas Gibbins, editors, *Engineering Knowledge in the Age of the Semantic Web, 14th International Conference, EKAW 2004*, volume 3257 of *Lecture Notes in Computer Science*, pages 63–81, Whittlebury Hall, UK, October 5–8th 2004. Springer.

14. Daniel Schober, Waclaw Kusnierczyk, Suzanna E Lewis, Jane Lomax, Members of the MSI, PSI Ontology Working Groups, Chris Mungall, Philippe Rocca-Serra, Barry Smith, and Susanna-Assunta Sansone. Towards naming conventions for use in controlled vocabulary and ontology engineering. In *The 10th Annual Bio-Ontologies Meeting, ISMB/ECCB*, 2007.
15. Rolf Schwitter, Kaarel Kaljurand, Anne Cregan, Catherine Dolbear, and Glen Hart. A Comparison of three Controlled Natural Languages for OWL 1.1. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC)*, Washington, 1–2 April 2008. 10 pages.
16. Paul R. Smart. Controlled Natural Languages and the Semantic Web. Technical Report ITA/P12/SemWebCNL, School of Electronics and Computer Science, University of Southampton, 2008.