Controlled Natural Languages for Knowledge Representation

Rolf Schwitter
Centre for Language Technology
Macquarie University
Rolf.Schwitter@mq.edu.au

Abstract

This paper presents a survey of research in controlled natural languages that can be used as high-level knowledge representation languages. Over the past 10 years or so, a number of machine-oriented controlled natural languages have emerged that can be used as high-level interface languages to various kinds of knowledge systems. These languages are relevant to the area of computational linguistics since they have two very interesting properties: firstly, they look informal like natural languages and are therefore easier to write and understand by humans than formal languages; secondly, they are precisely defined subsets of natural languages and can be translated automatically (and often deterministically) into a formal target language and then be used for automated reasoning. We present and compare the most mature of these novel languages, show how they can balance the disadvantages of natural languages and formal languages for knowledge representation, and discuss how domain specialists can be supported writing specifications in controlled natural language.

1 Introduction

Natural languages are probably the most expressive knowledge representation languages that exist; they are easy for humans to use and understand, and they are so powerful that they can even serve as their own metalanguages. Ironically, it is just this expressive quality that makes natural languages notoriously difficult for a computer to process and understand because a lot of relevant information is usually not stated explicitly in an utterance but only implied by the human author or speaker. There exist – of course – many useful resources and automated techniques that partly compensate for the lack of this background knowledge, and there are many useful applications that require only shallow processing of natural languages. But there exist – without doubt – many potential application scenarios that would benefit from deeper (axiom-based) knowledge that can be created and modified in a human-friendly way.

Formal languages (Monin, 2003) have been suggested and used as knowledge representation languages since they have a well-defined syntax, an unambiguous semantics and support automated reasoning. But these languages are often rather difficult for domain specialists to understand and cause a cognitive distance to the application domain that is not inherent in natural language.

One way to bridge the gap between a natural language and a formal language is the use of a controlled natural language (CNL) that can mediate between these languages. CNLs are engineered subsets of natural languages whose grammar and vocabulary have been restricted in a systematic way in order to reduce both ambiguity and complexity of full natural languages.

Traditionally, CNLs have been grouped into two broad categories: human-oriented CNLs and machine-oriented CNLs (Huijsen, 1998). The main objective of human-oriented CNLs is to improve the readability and comprehensibility of technical documentation (e.g. maintenance doc-

umentation (ASD Simplified Technical English[1]) and to simplify and standardise human-human communication for specific purposes (e.g. for trade or for air traffic control (see (Pool, 2006) for an overview)). The primary goal of machine-oriented CNLs is to improve the translatability of technical documents (e.g. machine translation (Nyberg and Mitamura, 2000)) and the acquisition, representation, and processing of knowledge (e.g. for knowledge systems (Fuchs et al., 2008) and in particular for the Semantic Web (Schwitter et al., 2008)).

Human- and machine-oriented CNLs have been designed with different goals in mind, and it is not surprising that their coverage can be quite different. O'Brien (2003) shows that there is not much overlap between the rule sets of CNLs in these two categories nor among the rule sets within a category. But since the structure of these CNLs is usually simpler and more predictable than the structure of full natural language, CNLs are in general easier for humans to understand and easier for a computer to process. An ideal CNL for knowledge representation should also be effortless to write and expressive enough to describe the problem at hand.

In this paper, we will survey machine-oriented CNLs that can be used for knowledge representation and can serve as high-level interface languages to knowledge systems. The rest of this paper is structured as follows: In Section 2, we introduce the most mature general-purpose CNLs and discuss the motivation for their design and investigate their characteristics. In Section 3, we discuss some theoretical issues regarding the expressivity and complexity of CNLs. Building on these theoretical considerations, we look in Section 4 at a number of machine-oriented CNLs that have been developed specifically as interface languages to the Semantic Web. In Section 5, we discuss the importance of supporting the writing process of CNLs in an suitable way and compare three different techniques. In Section 6, we discuss different approaches that have been used to evaluate the writability and understandability of CNLs, and finally in Section 7, we present our conclusions.

[1] http://www.asd-ste100.org/

2 General-Purpose CNLs

In this section we focus on a number of machine-oriented CNLs that have been designed to serve as knowledge representation languages. These CNLs are general-purpose languages in the sense that they have not been developed for a specific scenario or a particular application domain. These languages can be used where traditional formal languages are used otherwise. The aim of these languages is to equip domain specialists with an expressive knowledge representation language that is on the one hand easy to learn, use and understand and on the other hand fully processable by a computer.

2.1 Attempto Controlled English (ACE)

ACE (Fuchs et al., 2008) is a CNL that covers a well-defined subset of English that can be translated unambiguously into first-order logic via discourse representation structures (Kamp and Reyle, 1993) and then be used for automated reasoning. ACE is defined by a small set of construction rules that describe its syntax and a small set of interpretation rules that disambiguate constructs that might appear ambiguous in full English. The vocabulary of ACE consists of predefined function words (e.g. determiners, conjunctions, and pronouns), some predefined fixed phrases (e.g. *there is, it is false that*), and content words (nouns, proper names, verbs, adjectives, and adverbs). ACE supports language constructs such as:

- active and passive verbs (and modal verbs);
- strong negation (e.g. *no, does not*) and weak negation (e.g. *is is not provable that*);
- subject and object relative clauses;
- declarative, interrogative, imperative and conditional sentences;
- various forms of anaphoric references to noun phrases (e.g. *he, himself, the man, X*).

It is important to note that the meaning of words in ACE is not predefined; the user is expected to define their meaning by ACE sentences or import these definitions from an existing formal ontology.

Here is a simple example of an ACE text together with a question:

> Every company that buys at least three machines gets a discount. Six Swiss companies each buy one machine. A German company buys four machines. Who gets a discount?

Note that ACE uses disambiguation markers (e.g. *each*) on the surface level and mathematical background knowledge about natural numbers in order to answer the question above. This mathematical knowledge is implemented as a set of Prolog predicates which are executed during the proof (question answering process).

ACE is supported by various tools[2], among them a text editor that helps users to construct correct ACE sentences with the help of hints and error messages, a parser that translates ACE texts into discourse representation structures, a paraphraser that reflects the interpretation of the machine in CNL, and a Satchmo-style reasoning engine that can be used for consistency and redundancy checking as well as for question answering. Applications of ACE include software and hardware specifications, agent control, legal and medical regulations, and ontology construction.

2.2 Processable English (PENG)

PENG (White and Schwitter, 2009) is a CNL that is similar to ACE but adopts a more light-weight approach in the sense that it covers a smaller but fully tractable subset of English. The language processors of ACE and PENG are both based on grammars that are written in a definite clause grammar (DCG) notation. These DCGs are enhanced with feature structures and specifically designed to translate declarative and interrogative sentences into a first-order logic notation via discourse representation structures. In contrast to the original version of ACE that uses the DCG directly and resolves anaphoric references only after a discourse representation structure has been constructed, PENG transforms the DCG into a format that can be processed by a top-down chart parser and resolves anaphoric references during the parsing process while a discourse representation structure is built up. PENG has been designed for an incremental parsing approach and was the first CNL that was supported by a predictive editor (Schwitter et al., 2003). The PENG system provides text- and menu-based writing support that removes some of the burden of learning and remembering the constraints of the CNL from the user and generates a paraphrase that clarifies the interpretation for each sentence that the user enters. PENG's text editor dynamically enforces the grammatical restrictions of the CNL via look-ahead information while a text is written. For each word form that the user enters into the editor, a list of options is generated incrementally by the chart parser to inform the user about how the structure of the current sentence can be continued. The syntactic restrictions ensure that the text follows the rules of the CNL so that it can be translated unambiguously into the formal target language (first-order logic) and be processed by a theorem prover.

In order to illustrate how PENG can be used to reconstruct a problem in controlled natural language, we use an example from the TPTP problem library[3]. The problems in this library are usually used to test the capacity of automated reasoning tools and are translated manually by a human into the formal target language. For reasons of space, we use here one of the simpler problems of the library; the puzzle PUZ012-1 below is also known as "The Mislabeled Boxes":

> There are three boxes a, b, and c on a table. Each box contains apples or bananas or oranges. No two boxes contain the same thing. Each box has a label that says it contains apples or says it contains bananas or says it contains oranges. No box contains what it says on its label. The label on box a says "apples". The label on box b says "oranges". The label on box c says "bananas". You pick up box b and it contains apples. What do the other two boxes contain?

In order to solve this puzzle by a computer, we have to reconstruct it and augment it with the

[2] http://attempto.ifi.uzh.ch/site/tools/

[3] http://www.cs.miami.edu/~tptp/

relevant background knowledge. The main problems that we face here for machine-processing are the following ones: some of the constructions in the problem description are ambiguous (e.g. the antecedent for the personal pronoun *it* is open to two interpretations); the semantic relation between some content words is not explicit (e.g. the relation between the actual things in the box and the names on the labels that describe these things); and some of the constructions are not relevant at all for the solution of the problem (e.g. that the three boxes are on the table). Here is a possible reconstruction of this puzzle in PENG:

> The label of the box a says APPLES. The label of the box b says ORANGES. The label of the box c says BANANAS. APPLES stands for apples. ORANGES stands for oranges. BANANAS stands for bananas. All apples are fruits. All bananas are fruits. All oranges are fruits. Each box contains the apples or contains the bananas or contains the oranges. It is not the case that a box contains fruits and that the label of the box says something that stands for those fruits. It is not the case that a box X contains fruits and that a box Y contains those fruits. The box b contains the apples. What does the box a contain? What does the box c contain?

Note that this reconstruction makes information that is implicit or only assumed in the original problem description explicit in PENG.

PENG has recently been used for the construction of an interface to a situation awareness system (Baader et al., 2009) but the language can be used for similar applications to ACE.

2.3 Computer Processable Language (CPL)

CPL (Clark et al., 2010) is a controlled language that has been developed at Boeing Research and Technology. In contrast to ACE which applies a small set of strict interpretation rules, and in contrast to PENG, which relies on a predictive editor, the CPL interpreter directly resolves various types of ambiguities using heuristic rules for prepositional phrase attachment, word sense disambiguation, semantic role labeling, compound noun interpretation, metonymy resolution, and other language processing activities.

CPL accepts three types of sentences: ground facts, questions, and rules. In the case of ground facts, a basic CPL sentence takes one of the following three forms:

- There is|are *NP*
- *NP verb* [*NP*] [*PP*]*
- *NP* is|are *passive-verb* [by *NP*] [*PP*]*

Verbs can include auxiliaries and particles, and nouns in noun phrases can be modified by other nouns, prepositional phrases, and adjectives. In the case of questions, CPL accepts five forms; the two main forms are:

- What is *NP*?
- Is it true that *Sentence*?

In the case of rules, CPL accepts sentence patterns of the form:

- IF *Sentence* [AND *Sentence*]* THEN *Sentence* [AND *Sentence*]*

Parsing of CPL is performed bottom-up with the help of a broad coverage chart parser that uses preference for common word attachment patterns stored in a manually constructed database. During parsing, a simplified logical form is generated for basic sentences by rules that run in parallel to the grammar rules. There is no explicit quantifier scoping for these basic sentences and some disambiguation decisions (e.g., word sense and semantic relationships) are deferred and handled by the inference engine that makes a "best guess" of word sense assignments using WordNet[4]. The logical form is used to generate ground Knowledge Machine (KM) assertions. KM[5] is a frame-based language with first-order semantics. The KM interpreter employs a sophisticated machinery for reasoning, including reasoning about actions using a situation calculus mechanism. Rules

[4] http://wordnet.princeton.edu/

[5] http://userweb.cs.utexas.edu/users/mfkb/km.html

are entered by the user who writes CPL sentences with the help of rule templates. There exist seven templates for this purpose: three of them create standard logical implications and the rest describe preconditions and effects of actions. Each CPL sentence is interpreted interactively. The system paraphrases its interpretation back to the user, allowing the user to spot and fix misinterpretations. Sentences that express states add facts to a situation, and sentences that express actions trigger rules that update the situation, reflecting the changes that the action has on the situation. The user can ask questions about an emerging situation directly in CPL.

While CPL relies on heuristics, CPL-Lite is a slimmed down version of CPL that can be interpreted deterministically in a similar fashion to PENG. Each CPL-Lite sentence corresponds to a single binary relation between two entities. CPL-Lite distinguishes three types of relations: noun-like relations (e.g. the age of $<x>$ is $<y>$), verb-like relations (e.g. $<x>$ causes $<y>$), and preposition-like relations (e.g. $<x>$ is during $<y>$).

Interestingly, CPL-Lite has the same expressivity as CPL, but CPL-Lite is more verbose and grammatically more restricted. For example, the following two CPL sentences:

1. A man drives a car along a road for 1 hour.
2. The speed of the car is 30 km/h.

can be expressed (or better reconstructed) in an unambiguous way in CPL-Lite:

3. A person drives a vehicle.
4. The path of the driving is a road.
5. The duration of the driving is 1 hour.
6. The speed of the driving is 30 km/h.

Note that the user used here the noun *person* instead of *man* and *vehicle* instead of *car* during this reconstruction process because only these words were available in the system's ontology.

CPL and CPL-Lite have been mainly used to encode general and domain specific commonsense knowledge and to allow knowledge engineers to pose queries in a comprehensible way.

2.4 Other General-Purpose CNLs

Common Logic Controlled English (CLCE)[6] is a proposal for a CNL – similar to ACE and PENG – that has been designed as a human interface language for the ISO standard Common Logic (CL)[7]. However, CLCE itself is not part of this standard but uses Common Logic semantics. CLCE supports full first-order logic with equality supplemented with an ontology for sets, sequences, and integers. The primary syntactic restrictions are the use of present tense verbs, singular nouns, and variables instead of pronouns. Despite these limitations, CLCE can express the kind of English used in software specifications, mathematics textbooks, and definitions and axioms found in formal ontologies.

Formalized-English (Martin, 2002) is another proposal for a CNL that can be used as a general knowledge representation language. This language has a relatively simple structure and is derived from a conventional knowledge representation language. Formalized-English contains a number of formal-looking language elements and is therefore not a strict subset of standard English.

3 Theoretical Considerations

During the design of a CNL one has to pay attention to two important theoretical issues: the expressive power of the envisaged language and its computational complexity. E2V (Pratt-Hartmann, 2003) is a CNL that mainly grew out of theoretical studies about the expressivity and complexity of natural language fragments. E2V corresponds to the decidable two-variable fragment of first-order logic ($\mathcal{L}^2$). This fragment is interesting since it has the so-called finite model property. That means if a formula of $\mathcal{L}^2$ is satisfiable, then it is satisfiable in a finite model. E2V includes determiners (*every, no, a*), nouns, transitive verbs, verb phrase negation, relative, reflexive, and personal pronouns. Without any writing support it is difficult to decide if a sentence is in E2V or not. For example, one reading of sentence (7) is in E2V, the other one is not:

[6] http://www.jfsowa.com/clce/specs.htm
[7] ISO/IEC24707:2007

7. Every artist who employs a carpenter despises every beekeeper who admires him.

On the syntactic level, E2V is a subset of ACE with the exception that pronouns (e.g. *him*) always refer to the closest (acceptable) noun in the syntax tree (e.g. *artist*) and not to the closest (acceptable) noun that occurs in the surface structure (e.g. *carpenter*). This is because the E2V interpretation relies on the two-variable fragment of first-order logic. Note that sentence (7) has the following two possible representations (8 and 9) in first-order logic:

8. $\forall x_1$ (artist(x_1) & $\exists x_2$ (carpenter(x_2) & employ(x_1,x_2)) -> $\forall x_3$ (beekeeper(x_3) & admire(x_3,x_1) -> despise(x_1,x_3)))

9. $\forall x_1$ $\forall x_2$ (artist(x_1) & carpenter(x_2) & employ(x_1,x_2) -> $\forall x_3$ (beekeeper(x_3) & admire(x_3,x_2) -> despise(x_1,x_3)))

Although there are three variables in the formula (8) that correspond to the three nouns in sentence (7), the variables x_2 and x_3 never occur free in the same sub-formula. Therefore, the number of variables can be reduced by replacing x_3 through x_2. This technique can not be applied to the variables in formula (9).

E2V has been extended in various ways (Pratt-Hartmann and Third, 2006) and one extension includes counting determiners (e.g. *at least three, at most five, exactly four*). These determiners will not in general translate into the two-variable fragment of first-order logic, but into the fragment $\mathcal{C}^2$, which adds counting quantifiers to the two-variable fragment. The satisfiability problem of this fragment is still decidable and its expressivity and computational complexity is similar to those description logic languages that build the foundation of the Semantic Web.

4 CNLs for the Semantic Web

Recently, a number of CNLs have been developed that can serve as front-end to those formal languages that are used in the context of the Semantic Web[8]. These CNLs can be used by domain specialists who prefer familiar natural language-like notations over formal ones for authoring and verbalising formal ontologies.

ACE View (Kaljurand, 2007) is a CNL editor that supports a defined subset of ACE that can be used as an alternative syntax for the Semantic Web languages OWL and SWRL. ACE View integrates two mappings: one from ACE to OWL/SWRL and one from OWL to ACE. These mappings are not bidirectional in a strict sense since the OWL to ACE mapping also covers OWL axioms and expression types that the ACE to OWL mapping does not generate.

Sydney OWL Syntax (SOS) (Cregan et al., 2007) is a proposal for a CNL that builds upon PENG and provides a syntactically bidirectional mapping to OWL-DL. SOS is strictly bidirectional: each statement can be translated into OWL functional-style syntax and vice versa. The bidirectional translation is achieved with the help of a definite clause grammar that generates the target notation during the parsing process. In contrast to ACE, syntactic constructs of OWL are always carried over one-to-one to SOS. Thus, semantically equivalent OWL statements that use different syntactical constructs are always mapped to different SOS statements.

Rabbit (Hart et al., 2008) is a CNL designed for a scenario where a domain expert and an ontology engineer work together to build an ontology. The construction process is supported by a text-based ontology editor. The editor accepts Rabbit sentences, helps to resolve possible syntax errors, and translates well-formed sentences into OWL. The semantics of some Rabbit constructs is controversial (e.g. exclusive interpretation of disjunction) and hard to align with the semantics of OWL.

Lite Natural Language (Bernardi et al., 2007) is a CNL based on Categorial Grammar; it has the same expressivity as the description logic DL-Lite. DL-Lite is a tractable fragment of OWL and has polynomial time complexity for the main reasoning tasks. DL-Lite is expressive enough to capture relational databases and UML (Unified Modeling Language) diagrams.

[8] http://www.w3.org/TR/owl2-overview/

CLOnE (Funk et al., 2007) is a CNL that is built on top of the natural language processing framework GATE[9]. CLOnE is a simple ontology authoring language that consists of eleven sentence patterns which roughly correspond to eleven OWL axiom patterns. It is unclear whether CLOnE can be extended in a systematic way to cover larger fragments of OWL.

The three controlled languages ACE, SOS, and Rabbit are compared in more detail in Schwitter et al. (2008). There exist three other CNL research streams that are closely related to the Semantic Web: CNLs for querying Semantic Web content (Bernstein and Kaufmann, 2006); CNLs for maintaining semantic wikis (Kuhn, 2009; Kuhn, 2010); and CNLs for describing rules and policies (De Coi et al., 2009).

5 Writing Support for CNLs

Writing a specification in CNL is not an easy task since the author has to stick to the rules of the controlled language. Writing in CNL is in essence a normative process that prescribes how humans should use language to communicate effectively with a computer in order to achieve a particular goal. The challenge here is to develop interface techniques that make the writing process as unobtrusive and effortless as possible. Three main techniques have been suggested to support the writing process of CNLs: the use of error messages, conceptual authoring, and predictive feedback.

Error messages seem to be the most obvious way to support the writing of a text in CNL, and many CNLs (among them (Clark et al., 2010; Fuchs et al., 2008)) use this technique. The user is supposed to learn and remember the restrictions of the CNL and then to write the text following the memorised rules. If the parsing process fails, then the CNL system tries to identify the cause of the error and provides one or more suggestions for how to fix the error. The problem with this technique is that the input might be an unrestricted sentence and a useful error message would require in the worst case knowledge of the sort that is needed for processing full natural language.

[9] http://gate.ac.uk/

Conceptual authoring (Power et al., 2009) is a technique that allows authors to edit a knowledge base on the semantic level by refining specific categories and properties that occur in CNL sentences via a hierarchy of menu options. The selection of an option by the author results in an update of the underlying model and triggers the generation of a new sentence that can then be further refined. This method relies on natural language generation techniques and makes the analysis of CNL sentences unnecessary. The problem with this technique is that it does not allow the author to specify new knowledge that is not already encoded in the knowledge base; it is basically a technique for knowledge authoring and visualization and does not provide an independent knowledge representation language.

Predictive feedback (Schwitter et al., 2003; Kuhn and Schwitter, 2008) is a technique that informs the authors during the writing process about the approved structures of the CNL. This technique relies on interfaces that are aware of the grammar and can look-ahead within this grammar. Using this technique the author receives immediate feedback while a text is written and cannot enter sentences that are not in the scope of the grammar. The grammar of the language PENG has been designed from the beginning to be used in a predictive editor and is processed by a chart parser that is able to generate the look-ahead information. The following example illustrates how a predictive editor works:

- A [adjective | common noun]
- A man [verb | who | 'does not']
- A man works ['.' | preposition | adverb]

In this example the look-ahead information consists of syntactic categories, word forms and punctuation marks; all these elements are implemented as hypertext links. Selecting a hypertext link for a syntactic category displays approved word forms and selecting a word form or a punctuation mark directly adds this element to the text. Kuhn (2010) shows in an number of experiments that predictive editors are easy for untrained users to use and argues that predictive feedback is the best way to support the writing process of CNLs.

6 Evaluating CNLs

Over the past years, a number of different user experiments have been designed to measure various usability aspects of CNLs (see (Kuhn, 2010) for an introduction). These experiments can be grouped into three different categories: task-based experiments, paraphrase-based experiments, and graph-based experiments.

In task-based experiments (for example, (Kaufmann and Bernstein, 2007)), human subjects receive a certain task that requires them to use a CNL as an interface language to a knowledge base together with a tool that potentially supports the writing process. These experiments test how easy or difficult it is to write in these controlled languages using the given tool, but they do not test the understandability of these languages.

Paraphrase-based experiments (for example, (Hart et al., 2008)) aim to evaluate the understandability of a CNL in a tool-independent way. Human subjects receive a statement in CNL and a choice of paraphrases in full natural language, and then have to select the correct paraphrase. These experiments scale well with the expressivity of the CNL but it is difficult to guarantee that the paraphrases are understood in the intended way.

Graph-based experiments (for example, (Kuhn, 2010)) try to overcome the problems of paraphrase-based experiments. In order to test the understandability of CNLs and formal languages, a graph-based notation is used to describe a situation accompanied with statements in the language to be tested. The human subjects have to decide which of these statements are true and which ones are false with respect to the situation illustrated by the graph notation.

The reported results of these experiments in the literature provide strong evidence that CNLs are easier to write and easier to understand for domain specialists than formal languages.

7 Conclusions

It is an exciting time to work on controlled natural languages. In this paper, we surveyed a number of machine-oriented controlled natural languages that can be used instead of formal languages for representing knowledge. These controlled natural languages look like English but correspond to a formal target language. Anyone who can read English has already the basic skills to understand these controlled natural languages. Writing a specification in controlled natural language is a bit harder: it requires that the author either learns the language in order to be able to stay within its syntactic and semantic restrictions or that he uses an intelligent authoring tool that supports the writing process and enforces the restrictions of the language.

Machine-oriented controlled natural languages can be translated automatically (and often deterministically) into a formal target language (e.g. into full first-order logic or into a version of description logics). These languages can be used to express the kind of information that occurs in software specifications, formal ontologies, business rules, and legal and medical regulations.

In summary, an ideal machine-oriented controlled natural language should fulfill at least the following requirements: (a) it should have a well-defined syntax and a precise semantics that is defined by an unambiguous mapping into a logic-based representation; (b) it should look as natural as possible and be based on a subset of a certain natural language; (c) it should be easy for humans to write and understand and easy for a machine to process; and (d) it should have the necessary expressivity that is required to describe a problem in the respective application domain.

Of course these requirements can be in conflict with each other and therefore careful compromises need to be made when a new controlled natural language is designed. This design process offers many interesting research challenges for researchers in the area of computational linguistics and artificial intelligence. This research is driven by the overall goal to close the gap between natural and formal languages and to allow for true collaboration between humans and machines in the near future.

Acknowledgments

I would like to thank to three anonymous reviewers of Coling 2010 for their valuable feedback and to Robert Dale for comments and suggestions on previous versions of this paper.

References

Baader, Franz, Andreas Bauer, Peter Baumgartner, Anne Cregan, Alfredo Gabaldon, Krystian Ji, Kevin Lee, Dave Rajaratnam and R. Schwitter. 2009. A Novel Architecture for Situation Awareness Systems, In: *Proceedings of TABLEAUX 2009*, LNAI 5607, pp. 77–92.

Bernardi, Raffaella, Diego Calvanese, and Camilo Thorne. 2007. Lite Natural Language. In: *Proceedings of IWCS-7*.

Bernstein, Abraham and Esther Kaufmann. 2006. GINO – a guided input natural language ontology editor. In: *Proceedings of ISWC 2006*, LNCS 4273, pp. 144–157.

Clark, Peter, Phil Harrison, William R. Murray, and John Thompson. 2010 Naturalness vs. Predictability: A Key Debate in Controlled Languages. In: *Proceedings 2009 Workshop on Controlled Natural Languages (CNL'09)*.

Cregan, Anne, Rolf Schwitter, and Thomas Meyer. 2007. Sydney OWL Syntax – towards a Controlled Natural Language Syntax for OWL 1.1. In: *Proceedings of OWLED 2007*, CEUR, vol. 258.

De Coi, Juri L., Norbert E. Fuchs, Kaarel Kaljurand, Tobias Kuhn. 2009. Controlled English for Reasoning on the Semantic Web. In: *LNCS*, vol. 5500, pp. 276–308.

Fuchs, Norbert E., Kaarel Kaljurand, and Tobias Kuhn. 2008. Attempto Controlled English for Knowledge Representation. In: *Reasoning Web*, LNCS, vol. 5224, pp. 104–124.

Funk, Adam, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham, Brian Davis, and Siegfried Handschuh. 2007. CLOnE: Controlled Language for Ontology Editing. In: *Proceedings of ISWC 2007*.

Hart, Glen, Martina Johnson, and Catherine Dolbear. 2008. Rabbit: Developing a controlled natural language for authoring ontologies. In: *Proceedings of ESWC 2008*, LNCS, vol. 5021, pp. 348–360.

Huijsen, Willem-Olaf. 1998. Controlled Language – An Introduction. In: *Proceedings of CLAW 98*, pp. 1–15.

Kaljurand, Kaarel. 2007. Attempto Controlled English as a Semantic Web Language. *PhD Thesis*. Faculty of Mathematics and Computer Science, University of Tartu.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

Kaufmann, Esther and Abraham Bernstein. 2007. How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users? In: *Proceedings of ISWC/ASWC 2007*, NLCS, vol. 4825, pp. 281–294.

Kuhn, Tobias and Rolf Schwitter. 2008. Writing Support for Controlled Natural Languages. In: *Proceedings of ALTA 2008*, pp. 46–54.

Kuhn, Tobias. 2009. How controlled English can improve semantic wikis. In: *Proceedings of SemWiki 2009*, CEUR, vol. 464.

Kuhn, Tobias. 2010. Controlled English for Knowledge Representation. *Doctoral Thesis*. Faculty of Economics, Business Administration and Information Technology of the University of Zurich.

Martin, Philippe. 2002. Knowledge representation in CGLF, CGIF, KIF, Frame-CG and Formalized-English. In: *Proceedings of ICCS 2002*, LNAI, vol. 2393, pp. 77–91.

Monin, Jean-François. 2003. *Understanding Formal Methods.* Springer-Verlag, London.

Nyberg, Eric H. and Teruko Mitamura. 2000. The KANTOO Machine Translation Environment. In: *Proceedings of AMTA 2000*, LNCS, vol. 1934, pp. 192–195.

O'Brien, Sharon. 2003. Controlling controlled english – an analysis of several controlled language rule sets. In: *Proceedings of EAMT-CLAW 03*, Dublin City University, Ireland, pp. 105–114.

Pool, Jonathan. 2006. Can Controlled Languages Scale to the Web? In: *Proceedings of the 5th Int. Workshop on Controlled Language Applications*.

Power, Richard, Robert Stevens, Donia Scott, and Alan Rector. 2009. Editing OWL through generated CNL. In: *Pre-Proceedings of the Workshop on CNL 2009*, CEUR, vol. 448.

Pratt-Hartmann, Ian. 2003. A two-variable fragment of English. In: *Journal of Logic, Language and Information*, 12(1), pp. 13–45.

Pratt-Hartmann, Ian and Allan Third. 2006. More fragments of language: the case of ditransitive verbs. In: *Notre Dame Journal of Formal Logic*, 47(2), pp. 151–177.

Schwitter, Rolf, Anna Ljungberg, and David Hood. 2003. ECOLE – A Look-ahead Editor for a Controlled Language. In: *Proceedings of EAMT-CLAW03*, pp. 141–150.

Schwitter, Rolf, Kaarel Kaljurand, Anne Cregan, Catherine Dolbear, and Glen Hart. 2008. A comparison of three controlled natural languages for OWL 1.1. In: *Proceedings of OWLED 2008*, CEUR, vol. 496.

White, Colin and Rolf Schwitter. 2009. An Update on PENG Light. In: *Proceedings of ALTA 2009*, pp. 80–88.