An Update on PENG Light

Colin White
Department of Computing
Macquarie University
Sydney NSW 2109, Australia
colcwhite@gmail.com

Rolf Schwitter
Centre for Language Technology
Macquarie University
Sydney NSW 2109, Australia
Rolf.Schwitter@mq.edu.au

Abstract

This paper presents an update on PENG Light, a lightweight and portable controlled natural language processor that can be used to translate a well-defined subset of English unambiguously into a formal target language. We illustrate by example of a Firefox extension that provides a simple interface to the controlled natural language processor how web pages can be annotated with textual information written in controlled natural language and how these annotations can be translated incrementally into first-order logic. We focus in particular on technical aspects of the controlled language processor and show in detail how look-ahead information that can be used to guide the writing process of the author is generated during the parsing process. Additionally, we discuss what kind of user interaction is required for processing unknown content words.

1 Introduction

Computer-processable controlled natural languages are well-defined and tractable subsets of natural languages that have been carefully designed to avoid constructions that may cause ambiguities (Fuchs et al., 1998; Schwitter, 2002; Sowa, 2004; Barker et al., 2007). Instead of encoding a piece of knowledge in a formal language that is difficult to understand for humans, a controlled natural language can be used to express the same information in a direct way using the vocabulary of the application domain. There is no need to formally encode this information since a computer-processable controlled natural language can be translated automatically and unambiguously into a formal target language by a machine. This has the advantage that everybody who knows English can understand a text written in controlled natural language and that a machine can process this text since it corresponds to a formal notation. In order to support the writing of these texts, text- and menu-based predictive editing techniques have been suggested that guide the writing process of the author (Tennant et al., 1983; Schwitter et al., 2003; Thompson et al., 2005; Kuhn, 2008). These techniques give the author a way to match what he or she wants to express with the processing capabilities of the machine and result in user-friendly and self-explanatory interfaces. However, practically no details have been published so far how these predictive techniques can actually be implemented in a controlled natural language processor that processes a text incrementally. In this paper, we will make up for this neglect and show how these predictive techniques have been implemented for the controlled natural language processor of PENG Light and discuss how the language processor uses these techniques while communicating over an HTTP connection with a simple AJAX-based tool designed for annotating web pages in controlled natural language.

The rest of this paper is structured as follows: In Section 2, we give a brief overview of existing computer-processable controlled natural languages. In Section 3, we introduce FoxPENG, a simple Firefox extension that we have built for annotating web pages with controlled natu-

ral language and use this extension as a vehicle for motivating predictive editing techniques. In Section 4, we give a brief introduction to the controlled natural language PENG Light and show how sentences can be anaphorically linked and finally translated into discourse representation structures. In Section 5, we present the latest version of the chart parser of PENG Light and focus on the incremental processing of simple and compound words. In Section 6, we discuss what user interaction is required for processing unknown content words and suggest a linear microformat for specifying feature structures. In Section 7, we summarise the advantages of our controlled natural language approach for specifying a piece of knowledge.

2 Related Controlled Natural Languages

During the last decade, a number of computer-processable controlled natural languages have been designed and used for specification purposes, knowledge acquisition and knowledge representation, and as interface languages to the Semantic Web – among them Attempto Controlled English (ACE) (Fuchs et al., 1998; Fuchs et al., 2008), Processable English (Schwitter, 2002), Common Logic Controlled English (Sowa, 2004), and recently Boeing's Computer-Processable Language (Clark et al., 2005; Clark et al., 2007). Some machine-oriented controlled natural languages require the author to learn a small number of construction and interpretation rules (Fuchs et al., 2008), while other controlled natural languages provide writing support which takes most of the burden of learning and remembering the language from the author (Thompson et al., 2005). The commercial success of the human-oriented controlled natural language ASD Simplified Technical English (ASD, 2007) suggests that people can learn to work with restricted English and that good authoring tools can drastically reduce the learning curve of the language.

The language processors of ACE and PENG Light are both based on grammars that are written in a definite clause grammar (DCG) notation (Pereira and Shieber, 1987). 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 (Kamp and Reyle, 1993). In contrast to ACE that uses the DCG directly and resolves anaphoric references only after a discourse representation structure has been constructed, PENG Light 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.

3 The FoxPENG Toolbar

In order to annotate web pages with information that is at the same time human-readable and machine-processable, we developed FoxPENG, an AJAX-based Firefox extension (see Figure 1). FoxPENG supports the writing of annotations with the help of look-ahead information that indicates what syntactic categories or word forms can follow the current input. This look-ahead information is dynamically generated and updated by the language processor while an annotation is written. This application has some similarities to Google Suggest[1] and other autocomplete mechanisms[2] provided by source code editors, database query tools, and command line interpreters.

In contrast to Google Suggest that guesses what an author writes, the look-ahead information displayed in FoxPENG is a side-effect of the parsing process of the controlled natural language. FoxPENG communicates asynchronously with the language processor of PENG Light via a Prolog HTTP server[3] using JSON[4] (JavaScript Object Notation) as data-interchange format.

Once a connection to the Prolog server has been established, FoxPENG makes a request for the initial look-ahead categories to be displayed along the lower section of the toolbar. The author can now start typing an annotation that begins with a word form that falls under the corresponding look-ahead categories. Once a simple word form has been entered and the space bar has been pressed, an HTTP request is sent to the language processor containing the word form as well as the position of the word in the sentence.

[1] http://www.google.com/support/websearch/bin/answer.py?answer=106230

[2] http://en.wikipedia.org/wiki/Autocomplete/

[3] http://www.swi-prolog.org/packages/http.html

[4] http://json.org/

Figure 1: FoxPENG – Firefox Extension

The chart parser of PENG Light processes this information and either replies with a set of new look-ahead categories and word forms to choose from or with spelling suggestions in case of an error. The spelling suggestions are derived from the entries in the linguistic lexicon of PENG Light. If a content word is not misspelled and cannot be found in the linguistic lexicon, then the author can specify this word directly in the text area of FoxPENG using a microformat for linearised feature structures (see Section 6). An annotation can consist of more than one PENG Light sentence, and the language processor can resolve anaphoric references during the writing process using the standard accessibility constraints imposed by the discourse representation structures (see Section 4). The resulting discourse representation structure can be further translated into the input format of an automated reasoning engine and then be used for various reasoning tasks, among them for question answering. We have used the theorem prover and model builder E-KRHyper (Baumgartner et al., 2007) as reasoning service for PENG Light.

The author can publish a FoxPENG annotation as part of an RSS feed that contains a link to the annotated web page. In principle, any RSS feed aggregator can subscribe to such an RSS feed that is written in controlled natural language. This has the benefit that annotations are not only human-readable but also machine-processable (with the help of a PENG-compliant language processor).

4 PENG Light

PENG Light is a computer-processable controlled natural language that can be used for knowledge representation (Schwitter, 2008). At first glance, PENG Light **looks like** a subset of natural language, but PENG Light is actually a **formal** language since the language is designed in such a way that it can be translated unambiguously into a formal target representation. The vocabulary of PENG Light consists of predefined function words (determiners, coordinators, subordinators, prepositions and query words), a small number of predefined phrases (e.g. *there is*, *it is false that*) and content words (nouns, proper nouns, verbs, adjectives and adverbs) that can be defined by the author. A PENG text is a sequence of anaphorically interrelated sentences that consist of simple and complex sentences.

4.1 Sentences in PENG Light

PENG Light distinguishes between simple, complex, and interrogative sentences. Simple sentences consist of a subject and a verb (1-6), necessary complements (2-5), and optional adjuncts (5+6):

1. David Miller works.
2. David Miller teaches COMP249.
3. David Miller sends a letter to Mary.
4. David Miller is in the lecture hall.
5. David Miller teaches COMP249 on Monday.
6. David Miller works fast.

Complex sentences are built from simpler sentences through quantification (7+8), negation (9-11), subordination (12), and coordination (13):

7. Every professor teaches a unit.
8. David Miller teaches [exactly | at least | at most] two units.
9. If is false that a professor teaches COMP225.
10. No professor teaches COMP225.
11. David Miller is not a professor.

```
[drs([A, B, C],
  [theta(A, theme, C)#[1],
   event(A, working)#[1],
   theta(A, location, B)#[1],
   named(B, macquarie_university)#[1, [third, sg, neut],
                                    ['Macquarie','University']],
   named(C, david_miller)#[1, [third, sg, masc],['David','Miller']]])]
```

Figure 2: Annotated Discourse Representation Structure in PENG Light

12. David who teaches a unit supervises Mary.
13. David teaches COMP249 and supervises Mary.

A special form of complex sentences are conditionals (14) and definitions (15):

14. If David Miller works on Monday then Sue Rosenkrantz works on Tuesday.
15. A professor is defined as an academic who leads a research group and who teaches at least two units.

PENG Light distinguishes two types of interrogative sentences: *yes/no*-questions (16) and *wh*-questions (17):

16. Does David Miller teach a tutorial on Monday?
17. When does David Miller who convenes COMP249 teach a tutorial?

Interrogative sentences are derived from simple PENG Light sentences and serve the same purpose as queries in a formal query language.

4.2 Anaphora Resolution

In PENG Light, proper nouns, definite noun phrases and variables (that build an unambiguous alternative to pronouns) can be used anaphorically. The anaphora resolution algorithm of PENG Light resolves an anaphorically used noun phrase with the most recent accessible noun phrase antecedent that matches fully or partially with the anaphor and that agrees in person, number and gender with that anaphor. The anaphora resolution algorithm of PENG Light is embedded into the grammar and triggered whenever a noun phrase has been processed.

If a definite noun phrase can not be resolved by the anaphora resolution algorithm, it is interpreted as an indefinite noun phrase and introduces a new discourse referent into the universe of discourse.

4.3 Discourse Representation Structures

The language processor (DRS version) of the PENG Light system translates texts incrementally into TPTP notation (Sutcliffe and Suttner, 1998) with the help of discourse representation structures (DRSs) (Kamp and Reyle, 1993). The DRSs used in PENG Light rely on an event based notation (Davidson, 1967; Parsons, 1994) and a small number of thematic roles similar to (Kipper et al., 2008). Some of the conditions in the resulting DRS are annotated with syntactic information in order to improve the processability of the DRS by other system components (for example, the anaphora resolution algorithm). These conditions have the form `Pred#Anno` whereas `Pred` is a predefined predicate and `Anno` is a list that contains syntactic information. Figure 2 shows a simple DRS for the sentence (18):

18. David Miller works at Macquarie University.

The grammar of PENG Light contains not only feature structures for DRSs but also feature structures for syntactic and pragmatic information. In PENG Light, the construction of a DRS always runs in parallel with the construction of a syntax tree and a paraphrase. The paraphrase clarifies how the input has been interpreted by the grammar and can be used to show the author all relevant substitutions.

5 Chart Parsing in PENG Light

The grammar of PENG Light is specified in definite clause grammar (DCG) notation. However, the direct execution of a DCG would create many partial structures and destroy them while backtracking. This is not particularly efficient for generating look-ahead information (Kuhn and Schwitter, 2008). In order to avoid unnecessary repetition of work and to generate look-

ahead information efficiently, the DCG is transformed via term expansion (a well-known logic programming technique) into a notation that can be processed by the chart parser of PENG Light. The chart parser is based on work by (Gazdar and Mellish, 1989) but has been substantially extended to better support the incremental processing of PENG Light sentences, in particular to allow for generating look-ahead information, for processing compound words, and for resolving anaphoric references during the parsing process.

5.1 Basics of Chart Parsing

In general, a chart parser stores well-formed constituents and partial constituents in a chart (= table) that consists of a series of numbered vertices that are linked by edges (Kay, 1980). The vertices mark positions in the input and edges tell us what constituents have been recognised where for a given set of grammar rules. The chart parser of PENG Light represents edges as predicates with six arguments:

```
edge(SN,V1,V2,LHS,RHSFound,RHSToFind)
```

The first argument `SN` stores the sentence number, the subsequent two arguments state the existence of an edge between vertex `V1` and vertex `V2`, and the next three arguments represent information about the grammar rule. `LHS` is a category on the left-hand side of a grammar rule, `RHSFound` is a list of confirmed categories that have been found on the right-hand side of the grammar rule, and `RHSToFind` is a list of categories on the right-hand side that still need to be confirmed.

Inactive edges represent well-formed constituents where the right-hand side `RHSToFind` is empty, and active edges represent partial constituents where the right-hand side is not empty. The fundamental rule of chart parsing states what should happen when an inactive edge and an active edge meet. It specifies that whenever an inactive edge can extend an active edge, then a new edge (that is either active or inactive) is built and added to the chart. Finally, the prediction rule of chart parsing generates new active edges and is dependent on the first category on the right-hand side of a grammar rule and the previous state of the fundamental rule.

5.2 Initialising the Chart

We initialise the chart top-down and guarantee that a number of active edges are added to the chart using a failure-driven loop (see (Kuhn and Schwitter, 2008) for details). This initialisation process creates, for example, the following (radically simplified) set of active edges that start and end at vertex `0`for the first sentence:

```
edge(1,0,0,d(_),[],[s(_),pm(_)]).
edge(1 0,0,s(_),[],[np(_),vp(_)]).
edge(1,0,0,np(_),[],[det(_),noun(_)]).
edge(1,0,0,np(_),[],[pn(_)]).
edge(1,0,0,det(_),[],[lexicon(_)]).
edge(1,0,0,pn(_),[],[lexicon(_)]).
...
```

Additionally, the initialisation process adds the initial look-ahead information to the knowledge base. This look-ahead information consists of all those lexical categories (`lexicon(_)`) that occur as the first element in the list of unconfirmed categories and is stored in the following way in the knowledge base:

```
lookahead(FeatureStructures).
```

This makes it easy to extract the look-ahead information since the argument *`FeatureStructures`* consists of a list of feature-value pairs that contain the required syntactic and semantic information; additionally, the language processor can use this information to extract all word forms from the linguistic lexicon for a specific category that obey the current grammatical constraints.

5.3 Processing Simple Words

Once the chart has been initialised and a set of look-ahead categories has been displayed, the author can enter the first word form that belongs to one of these categories. Simple word forms are stored in the following way in the lexicon of PENG Light:

```
lexicon([
  cat:pn,
  wform:['John'],
  syn:[third, sg, masc],
  sem:[[I, person], atomic],
  con:named(I, john)]).
```

The chart parser uses the rule below together with Rule 4 in Figure 4 to look up a simple word (for example, *John*) in the lexicon:

```
(1)   add_edge(SN,V1,V2,LHS,Found,RHS) :-
        edge(SN,V1,V2,LHS,Found,RHS), !.

(2)   add_edge(SN,V1,V2,LHS,Found,[]) :-
        assert_edge(SN,V1,V2,LHS,Found,[]),
        apply_fundamental_rule(SN,V1,V2,LHS,[]).

(3)   add_edge(SN,V1,V2,LHS,Found,[RHS|RHSs]) :-
        assert_edge(SN,V1,V2,LHS,Found,[RHS|RHSs]),
        apply_fundamental_rule(SN,V1,V2,LHS,[RHS|RHSs]),
        predict_active_edges(SN,V2,RHS),
        update_lookahead_cats(SN,V2,RHS).
```

Figure 3: Add Edges

```
start_chart(SN,V1,V2,Word) :-
  foreach(word(T,SN,V1,V2,Word,LHS),
    add_edge(SN,V1,V2,LHS,Word,[])).
```

In a first step, the chart parser generates inactive edges for each occurrence of the word form in the lexicon using Rule 2 in Figure 3, after checking if such an edge does not already exist (Rule 1 in Figure 3). In the second step, the chart parser applies the fundamental rule of chart parsing recursively to inactive edges (Rule 2 in Figure 3) and active edges (Rule 3 in Figure 3). In the next step, the prediction rule of chart parsing is applied that looks for each grammar rule that has the category `RHS` previously used by the fundamental rule on the left-hand side (`LHS`), and generates new active edges for these categories. Once this has been done, the look-ahead information is updated (Rule 3 in Figure 3). For our example, this results in the following update of the chart:

```
edge(1,0,1,s(_),[np(_)],[vp(_)]).
edge(1,0,1,np(_),[pn(_)],[]).
edge(1,0,1,pn(_),['John'],[]).
edge(1,1,1,vp(_),[],[iv(_)])
edge(1,1,1,iv(_),[],[lexicon(_)])
...
```

Note that for each subsequent simple word form that the author enters, new grammar rules are triggered, new edges are added to the chart, and a new set of look-ahead categories is generated, extracted and then – in our case – sent to FoxPENG.

5.4 Processing Compound Words

As we have seen in the last section, the chart parser of PENG Light handles the input in an incremental fashion on a word by word basis. This creates problems for compound words. Each compound word is stored in the linguistic lexicon as a single entry of the following form:

```
lexicon([
  cat:noun,
  wform:[laptop,computer,bag],
  syn:[third,sg,neut],
  sem:[[I,entity],atomic],
  con:object(I,laptop_computer_bag)]).
```

This requires a special treatment of compound words by the chart parser since there are no grammar rules that describe the structure of these compound words, and a compound word can compete with other compound words or a simple word during processing. The chart parser of PENG Light uses three different rules (Rules 1-3 in Figure 4) to process compound words. The basic idea behind these rules is to retrieve each compound word only once from the linguistic lexicon as soon as the first element of a compound word becomes available and then maintain a store (`compound_word/6`) that is used to process all subsequent elements of the compound word (similar to edges). This will finally result in a single edge for the compound word in the chart.

Let us assume that the author is in the process of writing the compound noun *laptop computer bag*. After the first word (*laptop*) becomes available, the chart parser looks this word up in the linguistic lexicon using Rule 3 (and 4) in Figure 4, finds that this word is the first element of a compound word, and then checks if an active edge exists that corresponds to the category (`LHS`) on left-hand side of the grammar rule that

```
(1)   word(compound,SN,V1,V2,[Word],LHS) :-
        compound_word(SN,V0,V1,LHS,Found,[Word]),
        add_edge(SN,V0,V2,LHS,[Word|Found],[]).

(2)   word(compound,SN,V1,[Word],LHS) :-
        compound_word(SN,V0,V1,LHS,Found,[Word,LAH|LAHs]),
        edge(SN,V0,V0,LHS,[],[RHS|RHSs]),
        update_compound_word(SN,V0,V2,LHS,[Word|Found],[LAH|LAHs]),
        update_lookahead_cats(SN,V2,[LAH|LAHs]).

(3)   word(compound,SN,V1,[Word],LHS) :-
        call( LHS ==> [ lexicon([cat:Cat,wform:[Word,LAH|LAHs]|Rest],_) ] ),
        edge(SN,V1,V1,LHS,[],[RHS|RHSs]),
        call( lexicon([cat:Cat,wform:[Word,LAH|LAHs]|Rest],_) ),
        update_compound_word(SN,V1,V2,LHS,[Word],[LAH|LAHs]),
        update_lookahead_cats(SN,V2,[LAH|LAHs]).

(4)   word(simple,SN,V1,V2,[Word],LHS) :-
        \+ compound_word(SN,V0,V1,_,Found,[Word]),
        call( LHS ==> [ lexicon([cat:Cat,wform:[Word]|Rest],_) ] ),
        call( lexicon([cat:Cat,wform:[Word]|Rest],_) ).
```

Figure 4: Processing Simple and Compound Words

has been used for the lexicon lookup. In the next step, the chart parser updates the compound noun using the predicate `update_compound_word/6`. This predicate stores the sentence number (`SN`), the starting position (`V1`) and end position (`V2`) of the first element of the compound word, the category (`LHS`) on the left-hand side of the grammar rule, the found word (`[Word]`), and the remaining elements (`[LAH|LAHs]`) of the compound word. These remaining elements serve as new look-ahead information. If the author enters the next word (*computer*), the chart parser looks up this word in the store of compound words using Rule 2 in Figure 4, removes this word, checks if an active edge exists for this word, and then updates the store for compound words and the look-ahead categories. Finally, if the author enters the last element (*bag*) of the compound word, then the chart parser uses Rule 1 in Figure 4 and checks if the word is the last element of a compound noun, and adds a new edge to the chart that spans the entire compound word using Rule 2 in Figure 3, followed by a call to the fundamental rule. Note that this is a generic solution that can be used to process all categories of compound words.

6 Unknown Content Words

In principle, most function words can be displayed directly in the interface since there number is relatively small in controlled natural languages, but content words need to be structured in menus. These menus can be updated dynamically while a text is written. If the number of content words is large, then a copy of the linguistic lexicon can be loaded and maintained on the client side, and the task of the language processor is then reduced to inform the client about which categories of content words can follow the current input. In our case, PENG Light communicates with the client via a JSON object that has the following (simplified) form:

```
{"lookahead": [
    ["adj",["colour",
            "shape"]],
    ["noun",["masculine",
             "feminine",
             "masculine-feminine",
             "neuter-time",
             "neuter-entity"]] ],
 "paraphrase": ["A"] }
```

This object tells the client that either an adjective or a noun can follow the current input (in our case an indefinite determiner) and specifies

syntactic and semantic constraints for these categories. This information becomes also useful – as we will see below – if a content word is not available in the linguistic lexicon.

In real world applications, there are always cases where a required content word is missing from the linguistic lexicon. PENG Light allows the author to define a content word during the writing process. If the author enters a content word into the input field of the editor that is not yet defined in the lexicon, then the spelling corrector of PENG Light is used and provides alternative spellings that correspond to available lexical entries. PENG Light uses the Daumerau rules for this purpose that cover about 80% of human spelling errors (Damerau, 1964). These rules deal with the insertion, deletion, and substitution of single characters, and the transposition of two characters.

If a content word is not misspelled and not in the lexicon, then the author has to add the word form to the linguistic lexicon. The grammar already constrains the set of syntactic and semantic features that the author has to specify for a new content word. Let us assume that the word *laptop* is not yet in the lexicon. In this case the author has to specify only that this word is *neuter* and belongs to the sortal category *entity* but not that it is singular since this information can be derived from the current position of the word in the sentence and the information in the grammar. PENG Light accepts in-line specifications of linearised feature structures in a "microformat" notation, for example:

```
Input: A +n-n-e:laptop+
```

Note that this feature structure is an abbreviated notation that contains syntactic and semantic information derived from the feature structure provided by the JSON object. The plus symbol (+) functions as a control character that switches from the text entry mode to the vocabulary entry mode. The subsequent character sequence (*n-n-e*) represents the required feature structure, followed by a colon (:), the actual word form (*laptop*), and a plus symbol (+) that quits the vocabulary entry mode. The plus sign at the end of the word is necessary in order to deal with compound words, for example:

```
Input: A +n-n-e:laptop computer bag+
```

Using this approach, the author needs to specify only a minimal set of features and does not need to leave the text area in order to add a new content word to the user lexicon. For each category of content words, PENG Light maintains a list of unapproved words. A new content word is always checked against this list, before it is added to the user lexicon. Once a new word form has been successfully added to the linguistic lexicon, it is immediately parsed by the language processor and new look-ahead information is generated. Note that the author can only add new content words (adjectives, nouns, verbs and adverbs) using this microformat but not function words.

Similar to adding new content words, existing content words can be removed from the user lexicon. Alternatively, the microformat for feature structures is available from a menu of options.

7 Conclusions

In this paper, we presented an update on the controlled natural processor of PENG Light and showed how the language processor communicates with FoxPENG, an AJAX-based Firefox extension, using JSON as data-interchange format. For each approved word form that the author enters into the text area of this tool, the chart parser of PENG Light generates a set of look-ahead categories that determine what categories of word forms can follow the current input. This way only syntactically correct input is accepted by the language processor that can be translated unambiguously into a formal target notation. We focused in particular on an extension of the chart parser of PENG Light and showed in detail how compound words for which no grammar rules exist can be parsed incrementally during the writing process. We solved the unknown word problem with the help of a microformat for linearised feature structures that allows an author to specify unknown content words during the parsing process using minimal linguistic information. The controlled natural language PENG Light can be used as a high-level specification language for different kinds of knowledge systems and can help to solve the knowledge acquisition problem. Depending on the expressivity of the controlled language, the input can currently be translated into first-order logic or into a variant of description logic.

References

ASD 2007. ASD Simplified Technical English. *Specification ASD-STE100*, International specification for the preparation of maintenance documentation in a controlled language, Issue 4, January.

K. Barker, B. Agashe, S.-Y. Chaw, J. Fan, N. Friedland, M. Glass, J. Hobbs, E. Hovy, D. Israel, D.S. Kim, R. Mulkar-Mehta, S. Patwardhan, B. Porter, D. Tecuci, and P. Yeh. 2007. Learning by Reading: A Prototype System, Performance Baseline and Lessons Learned. In: *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pp. 280–286.

P. Baumgartner, U. Furbach, and B. Pelzer. 2007. Hyper Tableaux with Equality. In: *Proceedings of CADE-21*, LNAI 4603, pp. 492–507.

P. Clark, P. Harrison, T. Jenkins, T. Thompson, and R. Wojcik. 2005. Acquiring and Using World Knowledge Using a Restricted Subset of English. In: *Proceedings of FLAIRS'05*, pp. 506–511.

P. Clark, P. Harrison, J. Thompson, R. Wojcik, T. Jenkins, and D. Israel. 2007. Reading to Learn: An Investigation into Language Understanding. In: *Proceedings of AAAI 2007 Spring Symposium on Machine Reading*, pp. 29–35.

F.J. Damerau. 1964. A technique for computer detection and correction of spelling errors. In: *Communications of the ACM*, 7(3), pp. 171–176.

D. Davidson. 1967. The logical form of action sentences. In: Rescher, N. (ed.): *The Logic of Decision and Action*. University of Pittsburgh Press, pp. 81–95.

N.E. Fuchs, U. Schwertel, and R. Schwitter. 1998. Attempto Controlled English – Not Just Another Logic Specification Language. In: *Proceedings of LOPSTR'98*, pp. 1–20.

N.E. Fuchs, K. Kaljurand, and T. Kuhn. 2008. Attempto Controlled English for Knowledge Representation. In: *Reasoning Web, Fourth International Summer School 2008*, LNCS 5224, pp. 104–124.

G. Gazdar, C. Mellish. 1989. *Natural Language Processing in Prolog*. An Introduction to Computational Linguistics, Addison-Wesley.

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

M. Kay. 1980. Algorithm Schemata and Data Structures in Syntactic Processing. In: *CSL-80-12*, Xerox Parc, Palo Alto, California.

K. Kipper, A. Korhonen, N. Ryant, and M. Palmer. 2008. A large-scale classification of english verbs. In: Language Resources and Evaluation 42(1), pp. 21–40.

T. Kuhn. 2008. AceWiki: A Natural and Expressive Semantic Wiki. In: *Proceedings of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*.

T. Kuhn, R. Schwitter. 2008. Writing Support for Controlled Natural Languages. In: *Proceedings of ALTA 2008*, Tasmania, Australia, pp. 46–54.

T. Parsons. 1994. Events in the Semantics of English. A Study in Subatomic Semantics. MIT Press.

F.C.N. Pereira, S.M. Shieber. 1987. *Prolog and Natural-Language Analysis*. CSLI, Lecture Notes, Number 10.

R. Schwitter. 2002. English as a Formal Specification Language. In: *Proceedings of DEXA 2002*, September 2-6, Aix-en-Provence, France, pp. 228–232.

R. Schwitter, A. Ljungberg, and D. Hood. 2003. ECOLE – A Look-ahead Editor for a Controlled Language, In: *Proceedings of EAMT-CLAW03*, May 15-17, Dublin City University, Ireland, pp. 141–150.

R. Schwitter. 2008. Working for Two: a Bidirectional Grammar for a Controlled Natural Language. In: *LNAI 5360*, pp. 168–179.

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

G. Sutcliffe, C.B. Suttner. 1998. The TPTP Problem Library: CNF Release v1.2.1. In: Journal of Automated Reasoning, 21(2), pp. 177–203.

H.R. Tennant, K.M. Ross, R.M. Saenz, C.W. Thompson, and J.R. Miller. 1983. Menu-Based Natural Language Understanding. In: *Proceedings of the 21st Meeting of the Association for Computational Linguistics* (ACL), MIT Press, pp. 51–58.

C. Thompson, P. Pazandak, and H. Tennant. 2005. Talk to Your Semantic Web. In: *IEEE Internet Computing*, 9(6), pp. 75–78.