How to Write a Document in Controlled Natural Language

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

Anna Ljungberg
Centre for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
anna@ics.mq.edu.au

Abstract

This paper shows how a computer-processable document can be written in a controlled natural language (PENG) with the help of a sophisticated look-ahead editor (ECOLE). The editor provides syntactic hints after each word form entered and indicates how the author can continue the text. This way the author does not need to learn or to remember the restrictions of the controlled language. PENG documents are automatically translated into first-order logic via discourse representation structures. These formal entities can be checked by a theorem prover for inconsistency or consistency can be revealed by a model builder.

Keywords Document Processing, Controlled Languages, Authoring Tools.

1 Introduction

Documents such as software specifications or technical manuals are hard to write, as they need to be unambiguous and precise, the opposite to the qualities inherent in full natural language. Moreover, documents written in natural language are often hard to process automatically and inconsistency is difficult to detect. An alternative would be to write, for example, a software specification directly in a formal language. However, formal languages are hard to learn and to understand as they often abstract away from the facts of the application domain [4, 7].

A better strategy is to combine the advantages of a natural language with the advantages of a formal language and to hide the formality as far as possible from the authors. A controlled natural language, which is precisely defined by a restricted grammar and lexicon, can fulfill these requirements [1, 5, 6, 11, 12, 13].

PENG is such a controlled language and allows authors to write their texts in a well-defined subset of English, a language which is familiar to them [13]. To guarantee the painless and efficient usage of this language, ECOLE – a look-ahead editor – has been created and especially designed for PENG. The look-ahead editor guides the writing process and guarantees well-formed syntactic structures that can be translated into a formal language.

Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.

2 PENG by Example

PENG is a computer-processable controlled language specifically designed to write formal documents. PENG consists of a strict subset of standard English. The restrictions of the language are defined with the help of a controlled grammar and a controlled lexicon [13].

Let us illustrate the coverage of PENG by means of the Dreadsbury Mansion Mystery, a simple puzzle that is used in the literature [10] to test the capacity of automatic theorem provers. The puzzle is usually translated first by hand into a formal notation and the consequence of the puzzle is then proven by a theorem prover such as OTTER [9]. Using PENG, the manual translation becomes unnecessary, since PENG texts can be deterministically translated into first-order logic via discourse representation structures. Below follows the Dreadsbury Mansion Mystery in PENG:

> *A person lives in Dreadsbury Mansion. The person kills Agatha. Agatha is an aunt and lives in Dreadsbury Mansion. Agatha is a person. The butler lives in Dreadsbury Mansion. The butler is a person. Charles lives in Dreadsbury Mansion. Charles is a person. Every person that kills a person hates the person and is not richer than the person. If Agatha hates a person then Charles does not hate the person. If a person is not the butler then Agatha hates the person. If a person is not richer than Agatha then the butler hates the person. If Agatha hates a person then the butler hates the person. No person hates every person. Agatha is not the butler. Who kills Agatha?*

2.1 Controlled Lexicon

The lexicon of PENG consists of predefined function words such as

determiners: *a*, *the*, *every*, *no*, *who*, ...

prepositions: *in*, *than*, ...

copula: *is*

negation: *is not*, *does not*

coordinators: *and*, *or*

subordinators: *if ... then*, *before*, *after*, *while*

that build the structural scaffolding of the controlled language. User-defined content words such as

nouns: *butler*, *Agatha*, *Dreadsbury Mansion*, ...

verbs: *lives*, *kills*, *hates*, ...

adjectives: *rich*, *richer* ...

adverbs: *deeply*, ...

can be incrementally added or modified by the author during the writing process with the help of a lexical editor. Thus, by adding content words, the author creates his own application specific lexicon. In addition, the author can define synonyms for content words and acronyms or abbreviations for nouns. Illegal words (especially intensional words) can be defined in the lexicon by linguists.

2.2 Controlled Grammar

The controlled grammar defines the structure of simple PENG sentences and states how simple sentences can be joined into complex sentences by coordinators and subordinators. The grammar also specifies that simple sentences have by default a linear temporal order and that sentences can be interrelated in a well-defined way to build coherent texts. Simple PENG sentences are:

A person lives.

A person lives in Dreadsbury Mansion.

No person hates every person.

Agatha is a person.

Agatha is not the butler.

Complex PENG sentences are composed of simpler PENG sentences:

Every person that kills a person hates the person and is not richer than the person.

If Agatha hates a person then the butler hates the person.

3 ECOLE User Interface

The most important feature of ECOLE are the syntactic hints. After each word form entered, the author is given a list of choices of how to continue the sentence. These syntactic constraints ensure that the document remains unambiguous and precise. For example, when the author starts typing the sentence *The butler hates a person*, ECOLE displays the following look-ahead categories as subscripts in angle brackets:

The $_{[\ adjective\ |\ noun\ ]}$

The butler $_{[\ relative\ clause\ |\ verb\ |\ negation\ ]}$

The butler hates $_{[\ determiner\ |\ name\ ]}$ ...

As the example shows, the editor makes use of graphical means to display these syntactic hints as a help to the author. Not much linguistic knowledge is required to use this information as signpost. If something is unclear, then the author will be able to click on the displayed syntactic categories to get more information about that particular item.

The editor also handles compound nouns such as *Dreadsbury Mansion*. When the first noun *Dreadsbury* has been entered, the editor will respond with the second part of the compound noun *Mansion* and all other suitable look-ahead categories.

Another feature of ECOLE is the paraphrase that informs the author how the machine interpreted the input. Below follows an example of an input sentence and the paraphrase that is generated:

Input:

Agatha is not the butler.

Paraphrase:

Agatha is not [identical to] the butler.

The paraphrase thus makes it clear to the author that the copula (*is*) followed by a definite noun phrase (*the butler*) is interpreted as identity. If the copula had been followed by an indefinite noun phrase such as (*a butler*), then the machine would interpret this as a property and introduce a state.

PENG allows only well-defined forms of anaphoric references (definite descriptions and names, but no personal pronouns). The paraphrase displays how anaphoric references are resolved during parsing. An anaphoric expression is always replaced by the complete antecedent and the form is put within curly brackets. In PENG an anaphoric expression refers to the most recent accessible noun phrase that is suitable in terms of agreement, gender, and type, with respect to the nominal head and the pre- and postmodifiers.

Input:

A greedy butler lives in Dreadsbury Mansion.

The butler is a person.

Paraphrase:

A greedy butler lives in Dreadsbury Mansion.

{*The greedy butler*} *is a person.*

Another feature of the editor is the discourse representation structure (DRS) [8], which may be of interest to anyone wishing to see how the semantics of the information processed is represented. This feature can be used, for example, to teach students logic and computational semantics. A DRS captures the information in a multi-sentence discourse and shows the relations between the entities, the states, and the events in the application domain. From the following input the user will see the corresponding DRS:

Input:

Agatha kills the butler.

DRS:

```
[A,B,C]

named(A,agatha)

event(B,kill(A,C))

butler(C)
```

The first part of the DRS consists of a list of discourse referents `A, B, C` for the two individuals and the underlying event. These discourse referents are then used in the second part, which consists of conditions for the discourse referents. When a noun phrase is found to be anaphoric during parsing, it is directly resolved and not added to the DRS:

Input:

Agatha kills the butler. The butler is a person.

DRS:

```
[A,B,C,D,E]

named(A,agatha)

event(B,kill(A,C))

butler(C)

state(D,be(C,E))

person(E)
```

The example above shows that *butler* only appears once even though it is entered twice in the input sentence.

At first glance, PENG documents look informal and are easy to read, but they are in fact formal entities with all the nice properties of a formal language. Once translated into first-order logic these documents can be checked for inconsistency by a theorem prover or consistency can be revealed by a model builder [2].

4 How ECOLE Works

When the author types a word form into the editor then the current (partial) sentence is sent to the chart parser via a socket interface. The chart parser processes the input and generates the look-ahead categories. These syntactic categories are then displayed together with the paraphrase, the DRS, and the syntactic tree for the input for the options chosen by the author.

The grammar of PENG is implemented in the definite clause grammar (DCG) format. This unification-based approach allows us to resolve anaphoric references and to build up the DRS, the paraphrase, and the syntactic tree during parsing [3, 13]. Here is a typical grammar rule in DCG format:

```
n2( Agr, Index, Quant, Drs, Scope, ParaIn-
   ParaOut, [np,T1,T2], Gap-Gap, Ana)
   -->
   det( Agr, Index, Quant, Drs, Rest,
     Scope, ParaIn-Para, T1),
   n1( cat:cn, Agr, Index, Quant, Rest,
     Para-ParaOut, T2, Gap-Gap, Ana).
```

The chart parser processes such grammar rules top-down and produces edges according to the rules of chart parsing [5]. The edges have the following general form:

edge(START,END,HEAD,BODY)

Such edges simply tell us, what categories of a grammar rule ($HEAD \rightarrow BODY$) can span the substring of words found between the *START* point and the *END* point. We can distinguish two types of edges: active and inactive edges. An active edge is a hypothesis about a structure and an inactive edge is a result. For example, if the author types the determiner *the* into the editor, then the chart parser produces the following edges (simplified here):

```
edge(0,1,[det],[])

edge(0,0,[s],[n2,v2])

edge(0,0,[n2],[det,a2,n1])
```

```
edge(0,1,[n2],[a2,n1])

edge(1,1,[a2],[a1])

edge(1,1,[a1],[a0])

edge(0,0,[n2],[det,n1])

edge(0,1,[n2],[n1])

edge(1,1,[n1],[n0])
```

The first edge at the beginning of the chart is an inactive edge which contains an empty list `[]`. It represents a confirmed hypothesis and shows that a determiner has been parsed sucessfully between the nodes `0` and `1`. All other edges are active. That means that the chart is maintaining hypothesis about other structures that might follow.

The look-ahead categories are generated in the following way: During chart initialization the length `L` of the intput string is calculated and as soon as active edges are added to the chart that end at `L` then the leftmost category on the right hand side of a grammar rule is collected in a list. This results in two look-ahead trees from which the lexical categories `noun` and `adjective` can be easily derived.

5 Conclusions

This paper demonstrates how an unambiguous and precise document can be written in a computer-processable controlled natural language using a sophisticated look-ahead editor. Writing PENG puts no demands on the author when it comes to learn or to remember the rules of the controlled language as they are efficiently taken care of by ECOLE, the look-ahead editor.

The use of the look-ahead categories guarantees well-formed expressions and provides the necessary structural basis for the semantics of the controlled language in a completely compositional manner. PENG texts are deterministically translated into first-order logic and can be checked for consistency.

Acknowledgments

This research was supported by Macquarie University's New Staff Grant (MUNS 9601/0078). We would like to thank Mitko Razboynkov for developing the first version of the look-ahead editor, and David Hood for integrating the controlled grammar with the look-ahead editor.

References

[1] AECMA. 1988. The European Association of Aerospace Industries. *AECMA Simplified English*, AECMA Document PSC-85-16598. A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language. Issue 1, Revision 1, January.

[2] J. Bos. 2001. DORIS 2001: Underspecification, Resolution and Inference for Discourse Representation Structures. In Blackburn and Kohlhase (eds): *ICoS-3. Inference in Computational Semantics*. Workshop Proceedings, Siena, Italy, June.

[3] M. A. Covington, D. Nute, N. Schmitz, D. Goodman. 1988. From English to Prolog via Discourse Representation Theory. Research Report 01-0024. Artificial Intelligence Programs, University of Georgia.

[4] N. E. Fuchs, U. Schwertel, and R. Schwitter. 1999. Attempto Controlled English - Not Just Another Logic Specification Language. *Lecture Notes in Computer Science 1559*, Springer.

[5] G. Gazdar, C. Mellish. 1989. Natural Language Processing in PROLOG. An Introduction to Computational Linguistics, Addison-Wesley, Wokingham.

[6] C. Grover, A. Holt, E. Klein, and M. Moens. 2000. Designing a controlled language for interactive model checking. *Proceedings of the Third International Workshop on Controlled Language Applications*. 29-30 April 2000, Seattle, pp. 29–30.

[7] M. Jackson. 1995. *Software Requirements and Specifications, a lexicon of practice, principles and prejudices*. Addison-Wesley, Wokingham.

[8] H. Kamp and U. Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

[9] W. W. McCune. 1995. *Otter 3.0 Reference Manual and Guide*, Argonne National Laboratory, ANL-94/6, Revision A, August.

[10] F. J. Pelletier. 1986. Seventy-five Problems for Testing Automatic Theorem Provers, *Journal of Automated Reasoning 2*, pp. 191-216.

[11] S. G. Pulman. 1996. Controlled Language for Knowledge Representation. *Proceedings of the First International Workshop on Controlled Language Applications*, Katholieke Universiteit Leuven, Belgium, pp. 233–242.

[12] R. Schwitter. 1998. *Kontrolliertes Englisch für Anforderungsspezifikationen*. Dissertation, Institut für Informatik, Universität Zürich.

[13] R. Schwitter. 2002. English as a Formal Specification Language. *Proceedings of the Thirteenth International Workshop on Database and Expert Systems Applications (DEXA 2002)*, Aix-en-Provence, France, pp. 228-232.