Representing Knowledge in Controlled Natural Language: A Case Study

Rolf Schwitter

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

Abstract. In this case study I argue for the usage of a machine-oriented controlled natural language as interface language to knowledge systems. Instead of using formal languages that are difficult to learn and to remember for non-specialists, authors should be able to write specification texts in a well-defined subset of English that can be unambiguously processed by a computer. This subset of computer-processable English (PENG) consists of a restricted grammar and lexicon and is used together with an intelligent text editor that guides the writing process. The editor of the PENG system communicates with a language processor that generates logical structures while the author writes a specification text. The language processor is connected via a server with reasoning services that allow for acceptability checking and question answering. Texts written in PENG look seemingly informal and are easy to write and understand for humans but have first-order equivalent properties.

1 Introduction

Natural languages are the most powerful knowledge representation languages that exist. They can be easily learned in early childhood and have been optimised through evolution to support all aspects of human communication using one single notation. Natural languages serve as their own meta-language and have a greater flexibility and expressive power than any formal (artificial) language. However, natural languages used for writing precise specifications for knowledge representation have some serious disadvantages: they are difficult for computers to process because of their inherent ambiguity and vagueness, and their expressive power turns out to be one of the greatest obstacles for automatic reasoning [11].

Instead formal languages have been used to write specifications because these languages have an unambiguous syntax and clean semantics and help to avoid errors in the interpretation of the represented knowledge. Most of these formal languages are either directly or indirectly related to formal logic. However, such formal languages suffer from a number of unpleasant shortcomings: they are difficult to learn, difficult to use, and difficult to remember for non-specialists, since their notation very often abstracts away from the expressions used in a concrete application domain [2] [3].

Machine-oriented controlled natural languages [5] overcome most of these limitations and preserve the advantages of natural languages and formal languages. Some of these controlled natural languages have very interesting properties, since they seem informal at first glance but are formal languages with a precise syntax and semantics.

Different kinds of machine-oriented controlled languages have been developed to make multilingual machine translation of technical documents more efficient [9] and to make it feasible for machines to acquire and process knowledge expressed in a subset of natural language [10]. Provided that we support authors with an intelligent writing tool, even non-specialists can use such a controlled natural language to write specification texts for knowledge representation in a familiar notation without the need to formally encode the information [2].

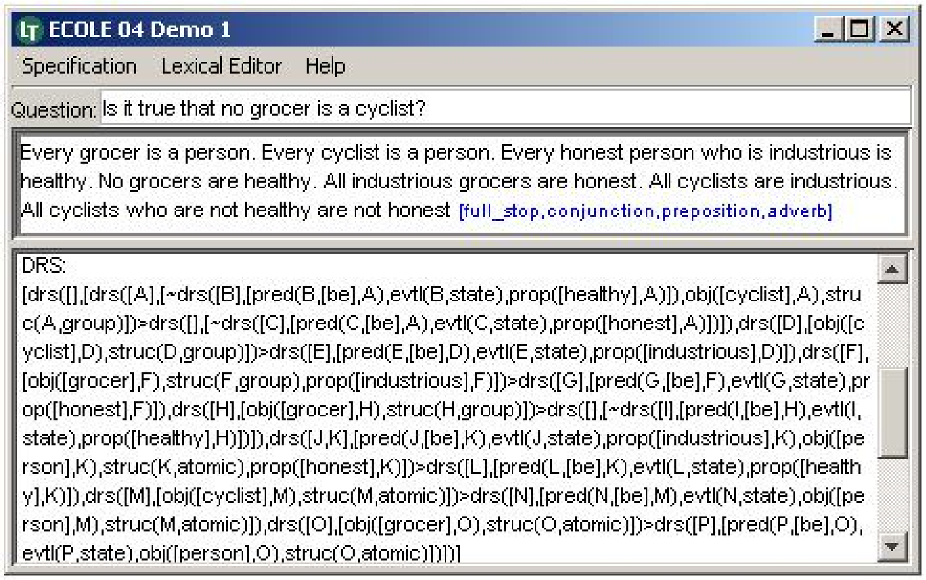
PENG is a machine-oriented controlled natural language that has been designed for non-specialists to write precise specification texts in a seemingly informal notation [12]. To guarantee the efficient usage of this controlled natural language, a text editor with an intelligent feedback mechanism has been developed that guides the writing process and guarantees well-formed linguistic structures that can be translated unambiguously into first-order logic via discourse representation structures (DRSs) [6]. The arising specification text can be checked on the fly for its acceptability constraints using third-party reasoning services. Additionally, the author can query a specification text in controlled natural language. In the case of PENG, the semantic representation of a specification is built up incrementally while the author writes the text and the interpretation of the machine is mediated by a paraphrase in controlled language.

2 PENG (Processable ENGlish)

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 [12] and are enforced by ECOLE, an intelligent text editor [13].

The controlled grammar defines the structure of simple sentences and states how simple sentences can be combined by coordinators (*and, or*) and subordinators (e.g. *before, after, if-then*) to build complex sentences. In PENG, the scope of quantifiers (e.g. *every, no, a*) can be determined from the surface order. A small number of constructors (e.g. *there is a/no, for every*) is available to change the relative scope explicitly on the surface level.

The controlled lexicon consists of predefined function words (*determiners, prepositions, coordinators, subordinators*), a set of illegal words (especially intensional words), and user defined content words (*nouns, verbs, adjectives, adverbs*). Content words can be incrementally added or modified by the author during the writing process with the help of a lexical editor (that is part of the intelligent text editor). Thus, by adding content words, authors create their own application specific lexicon. In addition, authors can define synonyms, acronyms, and abbreviations for content words.



Is it true that no grocer is a cyclist?

Every grocer is a person. Every cyclist is a person. Every honest person who is industrious is healthy. No grocers are healthy. All industrious grocers are honest. All cyclists are industrious. All cyclists who are not healthy are not honest [full_stop,conjunction,preposition,adverb]

```
DRS:
[drs([],[drs([A],[~drs([B],[pred(B,[be],A),evtl(B,state),prop([healthy],A)]),obj([cyclist],A),stru
c(A,group)])>drs([],[~drs([C],[pred(C,[be],A),evtl(C,state),prop([honest],A)])]),drs([D],[obj([c
yclist],D),struc(D,group)])>drs([E],[pred(E,[be],D),evtl(E,state),prop([industrious],D)]),drs([F],
[obj([grocer],F),struc(F,group),prop([industrious],F)])>drs([G],[pred(G,[be],F),evtl(G,state),pr
op([honest],F)]),drs([H],[obj([grocer],H),struc(H,group)])>drs([],[~drs([I],[pred(I,[be],H),evtl(I,
state),prop([healthy],H)])]),drs([J,K],[pred(J,[be],K),evtl(J,state),prop([industrious],K),obj([pe
rson],K),struc(K,atomic),prop([honest],K)])>drs([L],[pred(L,[be],K),evtl(L,state),prop([health
y],K)]),drs([M],[obj([cyclist],M),struc(M,atomic)])>drs([N],[pred(N,[be],M),evtl(N,state),obj([pe
rson],M),struc(M,atomic)]),drs([O],[obj([grocer],O),struc(O,atomic)])>drs([P],[pred(P,[be],O),
evtl(P,state),obj([person],O),struc(O,atomic)])])]
```

Fig. 1. The ECOLE editor with the *Grocer puzzle*, look-ahead categories and DRS

The text in Fig. 1 is a reformulation of Lewis Carroll's *Grocer Puzzle* in PENG. Note that ECOLE displays after each word form that the author enters look-ahead categories (e.g. *full_stop, conjunction, preposition, adverb*). These syntactic hints guide the writing process and constrain the subsequent input.

The main restrictions in the context of our puzzle are the use of present tense verbs and the control of plural constructions by disambiguation markers that reflect the interpretation in a paraphrase (e.g. *All cyclists [each] are industrious*). Other restrictions that are important are the scope of quantifiers and the scope of negation that can be determined from the surface order in PENG.

3 The Architecture of the PENG System

The top-level architecture of the PENG system consists of four main components (see Fig. 2): an intelligent text editor, a controlled language (CL) processor, a server, and reasoning services (consisting of a theorem prover and model builder). The text editor communicates with the CL processor via a socket interface. The CL processor is running as a client and is connected via the server with the reasoning services that are running separate client processes. The server implements a blackboard on which the CL processor writes a specification text (= theory) for which the theorem prover searches a proof and the model builder looks for a countermodel. These reasoning services are used to check the acceptability constraints (= consistency and informativeness) of a specification text and to answer questions about a specified piece of knowledge.

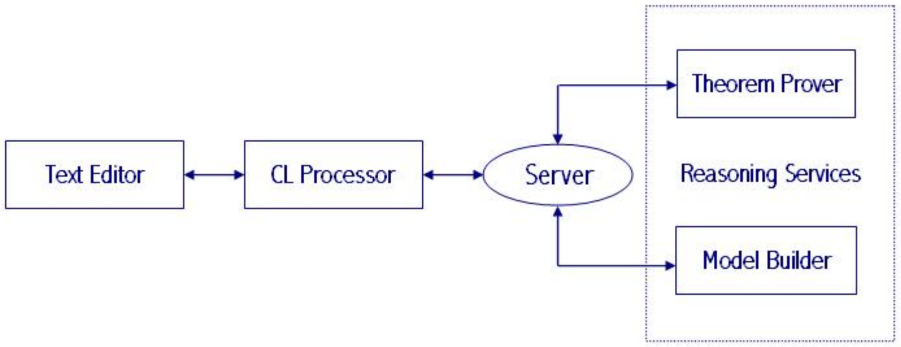


Fig. 2. Architecture of the PENG system

3.1 The Text Editor

The author interacts with the PENG system in controlled natural language using the text editor ECOLE and does not have to worry about the formal backbone of the system. Neither does the author need to know the grammar rules of the controlled language explicitly. ECOLE displays after each word form that the author enters what kind of syntactic structures can follow the current input (see Fig. 1). These look-ahead categories are generated on the fly while the text is written using the information produced by the chart parser of the CL processor [13].

The text editor comes with a spelling checker and an integrated lexical editor. If a content word is unknown and not misspelled, then the lexical editor pops up and allows the author to add the word to the lexicon. As soon as the word is available, the parsing process is resumed. If the corresponding options are selected, then the system checks the text for its acceptability constraints after each new sentence. If a new sentence violates these constraints, then the author gets immediate feedback. The author can also query a specification text in controlled language – in our case the author might be interested in finding out whether it is true or not that no grocer is a cyclist.

3.2 The CL Processor

After the author types a word form into the text editor, the token is immediately sent to the chart parser of the CL processor. The chart parser uses a unification-based (definite clause) grammar as syntactic scaffolding and constructs DRS conditions, look-ahead categories, information for a paraphrase, and resolves anaphoric definite references dynamically.

In our implementation a DRS is represented as a term of the form `drs(U,Con)` consisting of a list (U) of discourse referents $[I_1, I_2, \ldots I_n]$ denoting entities and a list (Con) of conditions $[C_1, C_2, \ldots C_n]$ that describe properties or relations that these discourse referents must satisfy. DRSs can occur as constituents of larger (complex) DRSs. Complex DRS conditions are those involving implication, disjunction, and negation (see also Fig. 1).

In contrast to Kamp & Reyle's original DRS construction algorithm [6], semantic information is threaded through grammar rules in PENG and a flattened notation for DRS conditions is used that treats concepts as typed individuals. Concepts do not introduce predicate symbols anymore and can therefore be referred to by simple terms (see also [4]). The domain of discourse in PENG is divided into the domain of objects and the domain of eventualities (= events and states). The domain of objects is a lattice-theoretic one and is subdivided into groups, individuals and mass objects.

Using a flattened notation has a number of advantages: First, quantification over complex terms that would require higher-order quantification can now be conducted via first-order quantification. Second, the flattened notation simplifies the formalization of logical axioms to express various forms of linguistic and non-linguistic knowledge. Third, this notation increases – as a neat side-effect – the efficiency of the inference processes.

3.3 The Reasoning Services

Standard reasoning services are not able to process DRSs directly. Therefore, a DRS that represents a (part of a) specification text is translated into a set of first-order formulas with the help of an efficient compiler that behaves linear on the size of the input [1]. These first-order formulas build a logical theory that can be investigated by a theorem prover (OTTER; [8]) and a model builder (MACE; [7]) that run in parallel.

Acceptability Constraints In PENG, we are especially interested in checking the acceptability constraints of a theory finding out whether the theory is consistent and informative after new information has been added to that theory. For example, if the author writes

All cyclists are industrious.

and later accidentally adds the information

No bikers are industrious.

then the consistency of the theory is violated. As we will see below, the PENG system can detect such inconsistencies provided that *cyclists* and *bikers* are stored as synonyms in the lexicon. In a similar way, if the author writes

Every cyclist is a person.

and later adds the information

Every biker is a person.

then the informativeness constraint is violated, since the second sentence does not add any new information to the specification text. Here, we would end up with a theory that contains redundant information.

Checking Acceptability Constraints To detect the inconsistency of a theory Φ, we can use a theorem prover such as OTTER and give it the negation of the theory $\neg\Phi$. If a proof is found for the negated theory, then the original theory is *inconsistent* (or unsatisfiable). To detect the consistency of a theory Φ, we can use a model builder such as MACE. MACE is a program that takes a theory and tries to build a model $\mathcal{M}$ for that theory. This is done with an interpretation function $\mathcal{I}$ that systematically maps predicates and constants of the language to members of a domain $\mathcal{D}$. A theory Φ is *satisfiable* (or consistent) if the model builder can find at least one model $\mathcal{M}$ that satisfies all the formulas in the theory. In general, model builders are only able to construct finite models and require a parameter that constrains the domain size of the model. OTTER and MACE can help each other out checking for inconsistency and satisfiability. The same reasoning services can also be used to check a theory for its informativness. Testing whether a piece of information Ψ is new and informative with respect to its previous context Φ can be done by giving the theorem prover $\Phi \rightarrow \Psi$. If it finds a proof, then Ψ is *not informative.* The model builder can do a similar test, provided that we give it $\Phi \wedge \Psi$ and then $\Phi \wedge \neg\Psi$; if the model builder finds a model $\mathcal{M}$ in both cases, then Ψ is *informative.*

Question Answering A variation of the basic proof procedure can be used to answer questions formulated in PENG. During a proof with OTTER, variables can be bound explicitly to values by substitutions with the help of answer literals. These bindings can be interpreted as a question answering process. On the other hand, a model builder such as MACE constructs flat structures with no explicit quantification or Boolean operators and allows for looking up the answer(s) to a question in the model [1]. As discussed, a DRS needs to be first translated into a set of first-order formulas before it can be processed by the reasoning services. Apart from the DRS, the reasoning services use additional lattice-theoretic axioms for the inference tasks. For instance, the following axiom

```
   (all X Y (struc(X,atomic) & part_of(X,Y) -> struc(Y,group))).
```

is used in PENG to relate a noun phrase (e.g. *every X*) that introduces an atomic object into the domain to a noun phrase (e.g. *all Xs*) that introduces a group. This linguistic axiom is necessary to answer the question of the puzzle. Since OTTER is a refutation-based theorem prover, we need to feed it the negation of the original question *Is it true that no grocer is a cyclist?* so that the result (in our case the empty clause) can be deduced automatically.

4 Conclusions

In this case study I presented the controlled natural language PENG together with the PENG system that allow non-specialists to write and process precise and unambiguous specification texts for knowledge representation. This case study shows that PENG is *easy to write* for non-specialists with the help of a look-ahead editor that guides the writing process, *easy to read* for non-specialists in contrast to formal languages, and *easy to translate* into first-order logic via discourse representation structures in contrast to unrestricted natural language. PENG can serve as a high-level interface language to any kind of knowledge systems and improve the knowledge acquisition process as well as increase the transparency of the knowledge representation for various kinds of applications. In the future, I am planning to study decidable subsets of PENG that can be translated automatically into a variant of description logic.

Acknowledgements

This research project was kindly supported by the Australian Research Council (DP0449928). I would also like to thank Marc Tilbrook for his work on the look-ahead text editor ECOLE and two anonymous reviewers for their valuable comments.

References

1. Bos, J.: 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 (2001)
2. Fuchs, N. E., Schwertel, U., Schwitter, R.: Attempto Controlled English – Not Just Another Logic Specification Language. In: LNCS 1559, Springer (1999) 1–20
3. Hall, A.: Seven Myths of Formal Methods. IEEE Software. Vol. 48, No. 1, (1990) 67–79
4. Hobbs, J.R.: Discourse and Inference. Draft. USC Information Science Institute, Marina del Rey, California, November 3 (2003)
5. Huijsen, W.O.: Controlled Language – An Introduction. In: Proceedings of CLAW 1998. Pittsburgh, (1998) 1–15
6. Kamp, H., Reyle, U.: From Discourse to Logic. Dordrecht: Kluwer (1993)
7. McCune, W.: MACE 2.0 Reference Manual and Guide. ANL/MCS-TM-249. Mathematics and Computer Science Division, Argonne National Laboratory, Argonne (2001)
8. McCune, W.: OTTER 3.3 Reference Manual. ANL/MCS-TM-263. Mathematics and Computer Science Division, Argonne National Laboratory, Argonne (2003)
9. Mitamura, T.: Controlled Language for Multilingual Machine Translation (invited paper). In: Proceedings of MT Summit (1999).
10. Pulman, S.G.: Controlled Language for Knowledge Representation. In: Proceedings of CLAW 1996. Katholieke Universiteit Leuven, Belgium (1996) 233–242
11. Sowa, J. F.: Knowledge Representation – Logical, Philosophical and Computational Foundations. Brooks/Cole (2000)
12. Schwitter, R.: English as a Formal Specification Language. In: Proceedings of the Thirdteenth International Workshop on Database and Expert Systems Applications (DEXA 2002). Aix-en-Provence (2002) 228–232
13. Schwitter, R., Ljungberg, A., Hood, D.: ECOLE: A Look-ahead Editor for a Controlled Language. In: Proceedings of EAMT-CLAW03, Controlled Language Translation, May 15-17, Dublin City University (2003) 141–150