University of Brighton

ITRI-99-11

Dialogue as Knowledge Editing

Paul Piwek, Roger Evans and Richard Power

January, 1999

Information Technology Research Institute Technical Report Series

ITRI, Univ. of Brighton, Lewes Road, Brighton BN2 4GJ, UK
TEL: +44 1273 642900 EMAIL: firstname.lastname@itri.brighton.ac.uk
FAX: +44 1273 642908 NET: http://www.itri.brighton.ac.uk

Dialogue as Knowledge Editing

Paul Piwek, Roger Evans & Richard Power

Abstract

This paper introduces a new approach to human-machine dialogue. This approach is based on the idea that dialogue can be seen as a form of knowledge base editing. In a dialogue, it is the dialogue history which is being edited. We propose some constraints on editing dialogue histories which distinguish dialogue from knowledge base editing in general. We claim that our proposal yields both interesting conceptual insights into the nature of dialogue, and provides a starting point for building practically feasible dialogue systems. A first, partly functional, prototype of such a dialogue system has been developed. We provide a description of this system.

1 Introduction

In this paper we describe a new approach to human-machine dialogue. We demonstrate that this approach provides both a fresh view on the nature of dialogue and helps us to build practically feasible dialogue systems.

Central to our approach is the notion of a shared dialogue history. The idea that utterances both give rise to updates of this history and are dependent on this history (e.g., for the interpretation of ambiguous expressions, pronouns, etc.) is well established (see, e.g., Clark [1996]).

Our aim is to take the idea that the utterances in a dialogue depend on and affect a shared history quite literally. A dialogue is modelled as a two agents editing a knowledge base (i.e., adding, cutting, copying and pasting knowledge objects) which represents the dialogue history. For this purpose, we use the WYSIWYM knowledge editing technology (see Power *et al.* [1998]).

We show that a dialogue can be seen as a special type of two-person knowledge base editing. We provide a number of constraints on two-person knowledge base editing which distinguish a dialogue from other forms of knowledge editing, thus obtaining a new view on the nature of dialogue.

The approach that we describe has been developed in the context of the CLIME project.[1] The overall aim of this project is a generic web-based system for accessing legal information. We describe the application that is being developed within the CLIME project to illustrate the suitability of our dialogue model for web-based human-computer interaction.

We proceed as follows. In section 2, we describe WYSIWYM style knowledge base editing. Next, in section 3, we introduce our model of dialogue as knowledge base editing. In section 4, the application, which is an information system for maritime law, is introduced. Section 5 contains our conclusions.

2 WYSIWYM Knowledge Base Editing

Generally speaking, there are two ways of editing a knowledge base: by means of a command language and by means of direct manipulation. Let us first consider the use of command languages. A command language requires an interpreter which updates the knowledge base on the basis of the user's text. Ideally, the user should be able to edit the knowledge base by means of the language which s/he is most accustomed to, i.e., unrestricted natural language. Unfortunately, for the foreseeable future unrestricted text seems to be not feasible for practical applications, because natural language understanding from free text is not yet sufficiently reliable. Therefore, so-called controlled languages have been introduced (see, e.g., Fuchs and Schwitter [1996]; van der Eijck [1998]). But there is a downside to the use of controlled languages: a user will have to learn to write in the controlled language, which can involve a substantial amount of effort.

Alternatively, there are direct manipulation interfaces to knowledge bases. Usually, the knowledge base consists of a network, which is graphically presented to the user (see, e.g., Paley [1996]). The user can now directly cut objects from the network and insert objects into the network. Thus each action by the user has a direct semantic interpretation. Hence, no parsing and interpret-

[1] CLIME stands for Computerized Legal Information Management and Explanation. It is funded by the EC Esprit Programme under project number EP 25.414.

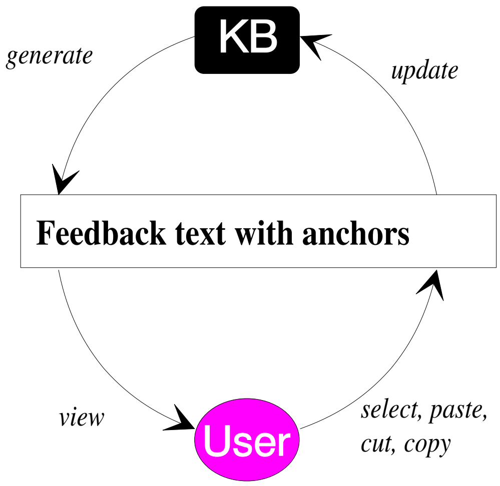

Figure 1:

ing is required. Again, there is also a downside. Network representations of knowledge can become very complicated and therefore difficult to understand by the user.

In Power *et al.* [1998], a new solution, called WYSIWYM (What You See is What You Meant), to the problem of knowledge editing has been proposed. WYSIWYM aims at the best of both worlds: the practical feasibility of direct manipulation and the ease of use of natural language text. The basic idea underlying WYSIWYM can be presented by means of a simple diagram, see Figure 1.

Figure 1. represents the editing cycle. Given a Knowledge Base (KB), the system generates a description of the knowledge base in the form of a 'feedback text' containing 'anchors' representing places where the knowledge base can be extended. Each anchor is associated with pop-up menus, which present the possible extensions of the KB at that point. On the basis of the extension that the user selects, the knowledge base is updated and a new feedback text is generated from the new contents of the KB. Additionally, spans of feedback text representing an object in the KB can also be selected by means of the mouse. Cut and copy operations are available which allow the user to cut or copy the underlying knowledge base object into a buffer. Subsequently, such an object can be pasted into a location where the KB is still incomplete. After a cut or paste action, a new feedback text is generated which represents the updated KB.

Let us consider a very simple example which allows us to briefly illustrate the essential features of WYSIWYM editing. As usual, the knowledge base consists of two parts: a T(erminological)-box and an A(ssertion)-box. In the T-box, we specify the set of available concepts and their attributes:[2]

(1) a. `domain > [like, person].`

[2]We use the ProFit notation. See Erbach [1995].

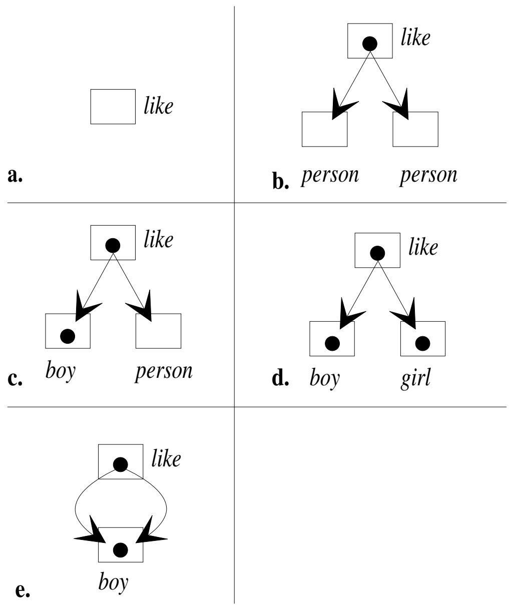

Figure 2:

b. `person > [man, woman, girl, boy].`
c. `like intro [attr1:person, attr2:person].`

We start with the concept `domain`, which has two subconcepts: `like` and `person` (1.a). Subsequently, we introduce the subconcepts of the concept `person` in (1.b). Finally, in (1.c), it is stated that the concept `like` has two attributes. A concept following a colon indicates which objects are legitimate values of the attribute.

The A-box contains the actual knowledge that is to be edited. Formally, this knowledge has the structure of a Directed Acyclic Graph (DAG). The nodes in the graph stand for the instances of concepts, i.e., objects, and the directed arcs of the graph represent attributes. The basic editing operation on a DAG is that of adding a new object, of a specified type, as the value of an attribute of an existing object. However, when we start knowledge editing, the A-box may be empty. To start the editing process, we need a predefined root attribute in the A-box, which is not part of the T-box proper, but whose value has to be an instance of one of the concepts that have been introduced in the T-box. For instance, let us assume that value of the root attribute has to be an instance of the concept `like`. This means that the state of the KB before we start editing can be represented with the graph in Figure (2.a). On the basis of this KB a feedback text is generated:

(2) **Some relation of liking.**

The entire span of text is in boldface. This indicates that the text is an anchor. By clicking on it, a menu appears which shows the possible alternatives for expanding the KB. In this case, the menu has only one option: `new object`. When the user selects this option a new object of the concept `like` is introduced into the KB (see fig. 2.b). On the basis of the new KB a fresh feedback text is generated:

(3) **Some person** likes **some person**.

The new feedback text has two anchors. When the user select the first anchor, a menu with the option `new object` appears again. But now, when user user clicks on `new object`, another menu appears, which allows the user to select the concept to which the object belongs, i.e., `man`, `woman`, `boy` or `girl`. Let us assume that the user selects `boy`. The new KB (fig. 2.c) gives rise to the following feedback text:

(4) A boy likes **some person**.

Along the same lines, the second anchor can be expanded. This leads to a fully completed knowledge base (fig. 2.d) and the following feedback text:

(5) A boy likes a girl.

Alternatively, instead of inserting a new object into the incomplete network (fig. 2.c), the user could also have chosen to copy an object that was already available, and paste it into the part of the network that was not yet developed. For instance, the user could have selected the span of text 'a boy'. A menu would have appeared with the options `cut` and `copy`. If the user had selected `copy`, the underlying object would have been stored in a buffer. Subsequently, the user could have selected the span of text which represents the incomplete part of the KB, i.e., '**some person**'. A menu would have appeared with the option `paste`. By performing the paste action, the network in fig. (2.e). would have been constructed, and the following feedback text would have been generated:

(6) A boy likes himself.

In this case, a reflexive pronoun is generated for the second attribute which takes the boy in question as a value. Note that if `copy` and `paste` had simply operated on the graphemic level of the sentence (i.e., the character sequence) instead of the underlying semantics, the result would have been 'A boy likes a boy'. For a more detailed account of coreference and WYSIWYM editing we refer to [van Deemter and Power, 1998].

3 Dialogue and Knowledge Base Editing

In a dialogue, two persons produce utterances and thus build up a shared history. This history is grounded in the physical events that constitute the utterances. On a higher level of analysis, the history consists of the words that were used and the syntactic structure of the phrases in which the words occur. Let us call this the linguistic level. Finally, on the highest level of analysis, the history consists of a sequence of speech acts. The speech acts each have a speech act type or force and a semantic content (Searle, 1969).

In ordinary dialogue the interlocutors edit the dialogue history by producing new utterances which extend the dialogue history. They also edit the dialogue history at its higher levels of analysis: by producing new utterances they also produce new speech acts. In this paper, we want to examine what it would be like if the interlocutors could directly edit the dialogue history at its highest level of interpretation: the speech act level.

WYSIWYM allows us carry out this experiment. For that purpose, we assume that the KB represent the dialogue history at the speech act level. Unrestricted two-person editing of this KB would, however, be nothing like ordinary dialogue. We need three simple constraints on the editing process to arrive at something which is similar to ordinary dialogue. We will state these constraints and then discuss their implementation in a WYSIWYM environment.

(1) *Turn taking*. In a dialogue, people normally speak one at a time. Therefore, we need to prevent the interlocutors from simultaneously editing the dialogue history. We thus also rule out that two participants simultaneously edit the same speech act.

(2) *Immutability of history*. Once something has been said, the fact that it has been said is immutable.[3] In terms of knowledge editing this means that only the most recent speech act can be changed in the editing process. Without this constraint in place we run into some serious difficulties. Suppose that it were allowed to edit speech acts further back in history. Note that some of the speech acts that occur after time some time T will depend on the speech acts before time T. This means that if a speech act before time T is edited, this also has repercussions for the ensuing speech acts. These repercussions can, however, not be overseen by one participant alone, since the ensuing speech acts were produced by both participants. For instance, if participant A were to change a question that preceded an answer produced by B, then this change would also affect the answer. Hence, the editing operation on the question would have no clearly delineated effect with respect to the entire dialogue history.

Note that although it is not possible to edit a dialogue history somewhere in the middle, this does not prevent us from going back to some point T in the history, and then pursuing a alternative history to the one that actually took place after T. This simply involves making a copy of the history from its starting point until T, and editing that copy (we shall see that this can be useful feature for practical dialogue systems).

(3) *Accessibility of history*. Although in ordinary dialogue, we cannot change previous utterances, we can make use of the content that they introduced, by means of, for instance, anaphora (e.g., 'A: The pump is not working. B: You have to replace *it*') and presuppositions ('A: The pump is out of order. B: Why *is it out of order*?')[4]. In terms of knowledge editing this means that

[3]Note that this does not mean that a participant can not contradict something that s/he said earlier on, but merely that the fact of having said it can not be 'undone'.

[4]The question 'Why P?' is said to presuppose that P.

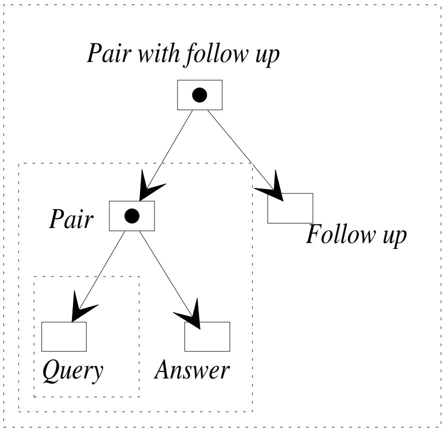

Figure 3:

we are allowed to copy items from the preceding history and paste them into the speech act that is currently under construction.

Let us now describe of how we can implement these ideas by means of a WYSIWYM system. The idea is that the interlocutors are a user and some software agent (e.g., an expert system). The user and the software agent talk with each other via a *dialogue manager* which maintains the dialogue history. The dialogue history is represented as a DAG. In this DAG, there will be nodes representing speech acts. At the beginning of a dialogue we start with DAG whose speech acts are not yet specified. In other words, the dialogue history still has to be fixed by the interlocutors. Now, the interlocutors can, one at a time, expand a speech act, thus step by step establishing a dialogue history.

Consider the DAG in figure 3. This DAG represents part of a dialogue history which still has to be filled in. There are locations representing three speech acts: a query, an answer and a follow up query to the answer (in speech act terms, we have two directives and one representative). The dotted boxes indicate in what order the interlocutors are allowed to edit the incomplete dialogue history. We start at the innermost box. Technically, this means that the root attribute is set to the location in the innermost box.

Let us assume that the user takes the first turn. The effect of having the `query` location as the value of the root is that the WYSIWYM generator will only generate a feedback text for this node. The feedback text will be something like: '**some query**'. The user can now start editing this query, until s/he is satisfied with the result (in the next section, we'll give an example of a partially expanded query). When the user has finished editing the query, s/he can hand over the turn to the other interlocutor. In practice, this will mean that the user interface will have a button for this purpose.

Now, it is the sofware agent's turn. The software agent will construct a representation of its answer (of course, it doesn't use the WYSIWYM interface, since the agent has been build in such a way that it can only construct well-formed representations). The software agent submits this answer to the dialogue manager, who slots it into the appropriate location of the dialogue history (the `answer` location).

Now, it is the user's turn again. At this point the root attribute is set to the `pair with follow up` location. Note that here the root is not set to an incomplete location, but rather to an object. The effect of this is that the generator will only generated the feedback text for the knowledge objects below this object. Thus a feedback text is generated which represents the full query and its answer, and an editable span of text for the follow-up query. At this point, the user can edit the `follow up` location and copy objects which are situated under the `pair` location into the follow up location. In other words, although the content of the preceding speech acts can no longer be edited, it is available for constructing the new speech act.

For the purpose of this example, we have identified the incomplete location of the dialogue history with very specific speech acts (query, answer, follow-up). We could also have chosen to use much broader types of speech acts as locations. For instance, we could have simply identified a location with a speech act of any type, and then let the user, by means of the WYSIWYM menu, choose whether it should be a question, conclusion, promise, declaration, etc. For a practical application, where the sequence of speech act types can be predicted reliably in advance, such freedom would, however, only have a negative effect on the ease of use of the system.

There is one limitation to the system as we have described it here. The range of possible speech act *types* that can be constructed at a given point in the dialogue does not depend on the preceding dialogue history (note, however, that the *content* of a speech act can depend on what has been said before, since the user is allowed to copy material from the preceding dialogue). This range is determined by the networks that the T-Box allows us to build. This T-box is static during the dialogue. We might, however, want to allow the user to only ask a follow-up question of a particular type, after an answer of a particular type. This can be achieved by giving the dialogue manager a more active role. In addition to simply moving the root attribute to the next incomplete speech act location, the dialogue manager might pre-edit this speech act on the basis of the information it has concerning the preceding dialogue, or simply hide certain, no longer relevant, ways to expand the location from the user.

For instance, suppose the system has just replied 'No', to the question 'Are there any pumps on this ship?'. In that case, a where-question ('Where are the pumps located') is no longer a sensible follow-up. Alternatively, if the answer had been 'Yes', a where-question would have been a natural follow up. Here it is useful if the system can dynamically decide whether to allow the user to construct a where-question or not.

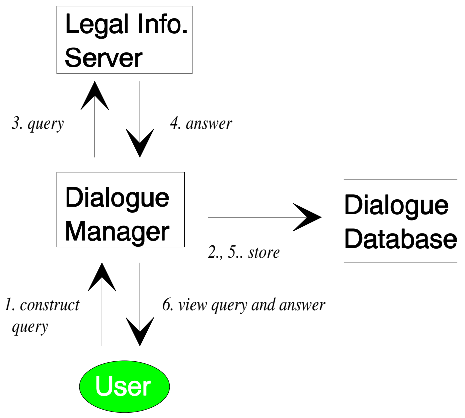

Figure 4:

This idea has, however, not yet been implemented in our first prototype.

4 An Application

The application that will be described in this section is being developed in the context of a project whose aim is to develop software to support access to legal and regulatory information. The concrete application that is under development is in the area of maritime law.

Let us first describe the context of use of the application. The potential end users of the application are the surveyors of a classification society. The business of this society is the production and application of rules, in particular, in the maritime domain. In order for a ship to be insured, the owner of the ship has to make sure that the ship has been certified by at least one such classification society. Certification is carried out by the surveyors, who check whether a ship fulfils all the rules that are laid down by the classification society.

Currently, the rules of the classification society are available both in paper form and on CD-Rom (with simple information retrieval functionality). One of our project partners has developed a legal reasoning system which can apply (a formalisation of) the rules to descriptions of ships. The system can, given a (partial) description of a ship, determine which rules apply to it, and whether the described situation is acceptable according to the rules or not.

The final system aims to help the surveyors in their daily practice of certifying ships. This means that the system should be available on site, for instance, during an inspection at a ship yard. Therefore, a web-based architecture has been chosen for the system. The user down loads the user interface as a JAVA applet into his/her computer, whereas the system itself is running on a possibly more powerful server elsewhere (legal information serving is computationally expensive for real life domains).

A simplified representation of the system architecture is depicted in Figure 4. The legal information server and the dialogue database are located on the server. The user interface to the dialogue manager is running on the user's computer as a client, whereas the main functionality provided by the dialogue manager is running on the server. The idea is that 1. the user constructs a query using WYSIWYM. 2. This query is stored in the dialogue database. 3. The query is submitted to the legal information server. 4. The legal information server returns back an answer, which is integrated into the dialogue history (up till that point it only contained the user's query) which the dialogue manager retrieves from the dialogue database. 5. Subsequently, the dialogue manager stores the updated dialogue history in a dialogue database. 6. The user can view (the feedback text corresponding to) the new dialogue history, and if needed continue to edit it.

There are two reasons for maintaining a dialogue database. Firstly, it allows the user to quickly construct queries which deviate not too much from queries which s/he submitted earlier on. For that purpose, the user simply retrieves a query from the database which is sufficiently similar to the one s/he wants to submits and modifies this query before submitting it. The second reason for maintaining a dialogue database is that this makes the dialogue independent of a persistent internet connection between the user's computer and the server. It is now possible to construct a follow up query to a query which was posted hours, days or even weeks ago. Furthermore, the legal information server may sometimes take more than an hour to compute an answer. It would be rather awkward if the user needed to remain logged in to wait for such an answer. Fortunately, this is not necessary because his or her query is stored in the dialogue database and updated with the answer whenever the legal information server has found one. The next time the user logs into the system s/he is notified of the change in the dialogue database (alternatively, the user may also be notified immediately by email) and can inspect the query and its answer.

We want to round off our description of the application by discussing a sample of the output that was generated by the actual system. The system is currently only a prototype. In particular, this means that not all of its parts have been integrated. However, the example we will discuss was generated by means of a fully functional WYSIWYM system.

The current system has a T-Box and a grammar which covers one chapter of the rules book of the classification society (there are 30 chapters in total). The example is modelled after the following query which was provided by experts from the classification society: 'I have a cargo ship with a gross tonnage of 500, that is fitted with a bilge pump with a capacity of 100m3/h in the main engine room. This pump is driven by the main engine and can provide water to the fire extinguishing system. Is this acceptable?'. Currently, a user can construct this query with the WYSIWYM system. The feedback text that is generated by the system is depicted in Figure 5 (the feedback text is not yet fully complete. Note that some phrases are printed in a lighter grey. These are anchors which indicate in which respects the underlying content of the query is still incomplete).

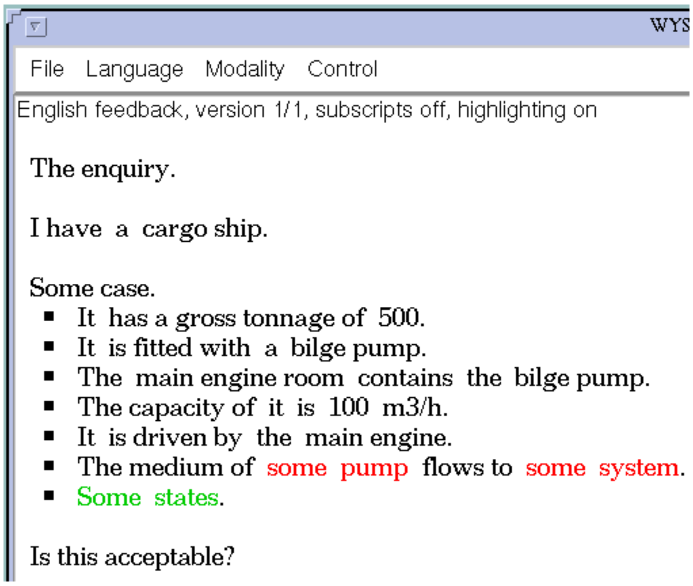

Figure 5:

Furthermore, on the basis of a formal representation of the answer to this query (which was handmade, because the legal information server and the dialogue manager have not yet been integrated, the text in Figure 6. is generated. In fact, the system generates a text from the representation of the query and the answer together (i.e., the dialogue history), in order to get the coreferences right (e.g., the use of 'the bilge pump' in the answer). In Figure 6., we also see an example of a follow-up enquiry.

This follow up query has been constructed starting from the following feedback text: 'Can you tell me why **some answer state**?'. The user can fill the location represented by the span '**some answer state**', by selecting a part of the answer, copying the underlying knowledge object, and pasting it into the follow up. Thus a complete follow up question is constructed.

5 Conclusions

- The model of dialogue as knowledge editing that we have introduced provides a fresh view on human-machine dialogue. In particular, by exploring the differences between dialogue and knowledge editing in general, we obtain a new perspective on the nature of dialogues.
- The model is promising for use in practical applications. It is based on the user friendly interaction provided by WYSIWYM editing. In combination with the use of a dialogue database, it has some features which make it particularly useful for human-machine dialogue over the internet.
- Currently, a demonstrator of the dialogue manager exists. It has not yet been integrated with the full application. However, the first trials with end-users of the system have been very encouraging. Three end users were asked to formulate queries using the WYSIWYM technology and had no difficulties in doing so.
- There is ample room for future research. Let us mention a few directions which we would like to pursue. Firstly, we would like to further explore the possibilities of making the range of utterances that the user can construct at a given point in the dialogue more dependent on the antecedent dialogue history. Furthermore, the issue of sub dialogues has not been addressed in this paper. In our application domain these can, however, not be avoided (e.g., sometimes the legal information system needs to obtain more background information from the user before it can answer a question). Finally, a formal evaluation of the application has been planned.

The answer.

Although the bilge pump is a firepump and it is driven by the main engine (see 20–052–11), the case is allowed, because the cargo ship has a gross tonnage of less than 1000 (see 20–052–31).

A follow up.

Can you tell me why the bilge pump is a firepump?

Figure 6:

6 References

[Clark, 1996] H. Clark. *Using Language*. Cambridge University Press, Cambridge.

[Erbach, 1995] G. Erbach. Prolog with features, inheritance and templates. In *Proceedings of EACL-95*, pages 180–197, Dublin. 1995.

[Fuchs and Schwitter, 1996] N. Fuchs and R. Schwitter. Attempto Controlled Language (ACE). In *First International Workshop on Controlled Language Applications*, Leuven, Belgium.

[Paley, 1996] S. Paley. Generic Knowledge-Base Editor User Manual. Technical Report, SRI International, California.

[Power *et al.*,1998] R. Power, D. Scott and R. Evans. What You See Is What You Meant: direct knowledge editing with natural language feedback. In *Proceedings of ECAI-98*, pages 677–681, Brighton, UK, 1998.

[Searle,1969] J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge.

[van der Eijck, 1998] P. van der Eijck, Controlled Languages in Technical Documentation. In *Selected Papers from the Eight CLIN meeting*, pages 187–203, Rodopi, Amsterdam.

[van Deemter and Power, 1998] K. van Deemter and R. Power, Coreference in knowledge editing. In *Proceedings of the COLING-ACL workshop on Computational Treatment of Nominals*, pages 56–60, Montreal, Canada, 1998.