Speaker-independent context update rules for dialogue management

Samson de Jager, Nick Wright, Alistair Knott
Dept. of Computer Science
University of Otago
sdejager/nwright/alik@cs.otago.ac.nz

Abstract

This paper describes a dialogue management system in which an attempt is made to factor out a declarative theory of context updates in dialogue from a procedural theory of generating and interpreting utterances in dialogue.

1 Background: declarative and procedural resources for text processing

In computational linguistics, a very useful distinction can be drawn between declarative models of language, which specify what constitutes a 'correct' or 'proper' sentence or discourse, and procedural models, which specify how a proper sentence or discourse can be interpreted or generated. This distinction is virtually ubiquitous in computational models of sentence structure and sentence processing. In these models, a sentence grammar is a declarative construct, and a sentence parser or sentence generator consults the grammar in a systematic way at each iteration in order to interpret or create sentences. The idea of systematicity is very important. For instance, in a chart parser the process of sentence parsing is broken down into a sequence of procedural operations each of which has exactly the same general form: a search of the set of grammar rules, and the creation of a new chart edge if the search is successful. In fact, the benefits of thinking of a parser as a procedural module consulting a declarative grammatical resource are most clearly seen in the fact that the procedural component can be expressed systematically in this way.

The declarative/procedural distinction is also increasingly common in computational treatments of extended monologues. There are several overtly declarative theories of the structure of such texts (many of them stemming from the work of Mann and Thompson (1988) and Grosz and Sidner (1986)), and several models of text generation and text interpretation which make reference to these declarative theories (see e.g. Hovy (1993) and Marcu (2000) for a summary of generation and interpretation methods respectively). Again, the most attractive feature of the declarative/procedural distinction is that the procedural algorithms envisaged are very systematic.

In models of dialogue structure, a clean separation between declarative and procedural models has proved more elusive. By analogy with the cases of sentences and monologic discourse just described, what is required is a declarative model of 'well-formed dialogue', together with procedural mechanisms that consult this model in a systematic way to produce and interpret contributions to a dialogue.

To begin with, what is a declarative theory of dialogue structure? In this paper, we will assume a theoretical perspective in which dialogue moves are represented as context-update operations (see e.g. Traum *et al* (1999)). An utterance in a dialogue is understood as a function which (when defined) takes the current dialogue context and outputs a new dialogue context that constitutes the input to the next utterance in the dialogue. The declarative theory of dialogue coherence will therefore be a theory about the legal ways in which the dialogue context can be

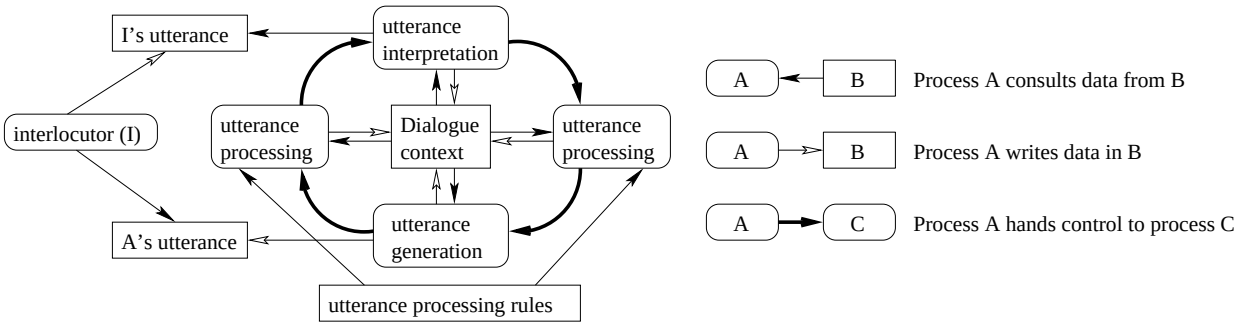

Figure 1: A model of dialogue management factoring out a declarative component of 'utterance processing'.

updated.[1] The question we focus on is: how can we formulate a model of dialogue *processing* that makes reference in a systematic way to this declarative theory?

The difficulty is that 'dialogue processing' is not a uniform activity. There are two very different processes involved: one is utterance interpretation, and the other is utterance generation. An agent participating in a dialogue runs these two processes roughly in alternation. If an algorithm for dialogue processing is to be systematic in the way discussed above, we expect it to consult the declarative theory of context updates in the same way when processing each utterance, whether it is being generated or interpreted. But is this really possible? The process of generating an utterance seems very different from that of interpreting one. Perhaps most obviously, when an agent is generating an utterance, it needs to work out what the content of this utterance is, while when the agent is interpreting an utterance, this content is handed to it on a plate, and it simply has to work out how it is to be incorporated into the current discourse context.

2 Dialogue management using speaker-independent resources

This paper describes the dialogue management scheme implemented in a dialogue system called Te Kaitito[2] (Knott et al., 2002; de Jager et al., 2002). In this scheme, an attempt is made to factor out a **speaker-independent** component of utterance processing which is the same whether the utterance is being generated or interpreted from a **speaker-dependent** component of utterance processing which is completely different for utterance generation and utterance interpretation. Both of these types of processing are thought of as operations which perform updates on the dialogue context. The basic idea is that a speaker-dependent operation is responsible for adding new information about an utterance and its semantic content to the dialogue context, while a speaker-independent operation consults a declarative model of dialogue structure to further process this information in the light of what is already in the dialogue context. Imagine an agent A participating in a dialogue with an interlocutor I. When A is generating an utterance, we propose that A performs two completely separate operations. Firstly there is a speaker-dependent operation of **utterance generation**, which is performed using a collection of very procedural and very domain-specific resources, including A's current plans and knowledge of the world. As a consequence of this operation, a new **utterance representation** is added to the dialogue context, and (as a side-effect) A actually makes an utterance. Secondly, a speaker-independent operation of **utterance processing** occurs, in which a set of **utterance processing rules** are consulted, and various further changes are made to the dialogue context. Imagine now that A interprets a response utterance coming from I. There are again two separate operations. Firstly there is a speaker-dependent operation of **utterance interpretation**, involving sentence parsing, syntactic and se-

[1] Note that there is nothing procedural in the notion of an update; it is just a way of characterising the meaning of an utterance in a dialogue.

[2] Te Kaitito is a bilingual system, which supports dialogues in either English or Māori. *Te Kaitito* means 'the extempore speaker' or 'the improviser'.

mantic disambiguation and so on. As a consequence of this operation, again a new utterance representation is added to the dialogue context. Secondly, another speaker-independent operation of **utterance processing** occurs, in which exactly the same set of utterance processing rules are consulted, and further context updates are made. This picture of dialogue processing is summarised in Figure 1. In the remainder of the paper, this picture will be picture will be examined in more detail.

3 A DRT-based model of utterances and dialogue context

The semantics of sentences in Te Kaitito are represented using Discourse Representation Structures (DRSs: see (Kamp and Reyle, 1993)), both for sentence generation and sentence interpretation. The main extension of classical DRT is to incorporate a treatment of presuppositions along the lines of that given in van der Sandt (1992). In this treatment, a sentence is modelled as an **assertion DRS** and a set of **presupposition DRSs**, each of which specifies information which must be found in the dialogue context before the sentence's assertion can be further processed. Our system is a modification of van der Sandt's, in that presuppositions are used to model the context-dependence of questions and answers as well as phenomena like definite descriptions and anaphora (see de Jager *et al.* (2002) for some motivation for this idea, and see Section 5 for some examples).

The dialogue context in Te Kaitito is also represented as a DRS, with additional extensions roughly along the lines of Poesio and Traum (1998). The main extension is the idea of a **discourse unit**, which is a sub-DRS gathering all of the semantic information expressed in a single utterance by one of the dialogue participants, the whole of which is treated as a first-class object to which other predicates can apply (e.g. $asserts(speaker, U1)$). This idea of discourse units is also adopted in the MIDAS system (Bos and Gabsdil, 2000). The other extension of classical DRT (also following Poesio and Traum) is that the dialogue context is modelled as a *pair* of DRSs: a **common ground DRS** containing facts which are mutually believed, and a **stack DRS** containing **ungrounded** information (questions which have not been fully answered, assertions which have not been acknowledged, and so on). The stack DRS is actually a sub-DRS of the common ground DRS in our implementation, capturing the idea that referents within the common ground are available within the stack DRS, but not vice versa.

Our model differs from that of Poesio and Traum in one fairly small respect, because we also envisage a role for discourse units in text planning. One of the functions of a dialogue in our system is to allow a user to author a knowledge base of facts that serves as input to a text planning system (Knott and Wright, 2003). A text planner needs to be able to partition its knowledge base into 'utterance-sized' sets of facts, which can then be structured into larger texts in various ways. It is very useful if the system can remember how the user performed these partitions during the authoring dialogue. The discourse units of Poesio and Traum seem almost tailor-made for this purpose. However, not all of the predicates inside a discourse unit should be retained in these cached utterances. Some predicates are only present in a sentence because they feature in referring expressions; since referring expressions will be different depending on the context in which they are produced, we need a separate context-specific routine to add these predicates to the semantics of an utterance when a sentence is to be produced. See Knott and Wright (2003) for more details about how these cached utterances are obtained and used.

An example of a dialogue context is given in Figure 2. This is how the context would appear after the

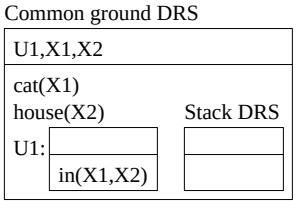

Figure 2: A simple dialogue context

interlocutor I has uttered the sentence *A cat was in a house*, and the agent A has successfully added this information to the common ground. Note first that the stack DRS is empty. Note also that the predicates $cat(X1)$ and $house(X2)$ have been moved from the utterance unit $U1$ into the top level of the common

ground, while the actual predication associated with this sentence stays in the utterance. This 'unloading' of material into the top-level DRS is superficially similar to the operation performed in MIDAS system for grounding an utterance. But in our case its function is purely to do with the text-planning application, and has nothing to do with making referents accessible. We assume all material inside utterance units is accessible as if it were in the outer DRS.

4 Utterance processing rules

In the top-level control loop of the dialogue manager described in Section 2, when an utterance is interpreted or generated (using speaker-specific processes), a new **utterance representation** is added to the stack DRS. An utterance representation consists of a discourse unit as described in Section 3, plus two kinds of predicate about the utterance represented by this unit: firstly a predicate specifying what dialogue act it performs, and secondly predicates specifying which variables it contains. After this comes the speaker-independent operation of utterance processing introduced in Section 2.

Utterance processing is modelled as a cyclical process of applying context update rules, in the kind of way envisaged in the MIDAS system (Bos and Gabsdil, 2000). An **utterance processing rule** contains a **condition** to look for in the dialogue context, and an **action**, which is an update operation on the context. At each cycle, the set of rules is searched, and the first rule whose condition matches has its action executed. The cycle is repeated until no rules trigger.

There is no reason *a priori* why speaker-independent context update operations need be applied cyclically by a rule interpreter. But there are several reasons why one might want to use a cyclical scheme. For Bos and Gabsdil (2000), the primary reason is that updating the context involves a great deal of reasoning, and can in fact be construed as a kind of reasoning. Reasoning is naturally modelled as the cyclical application of rules of inference, and hence a cyclical framework for updates seems quite natural. In our system, however, the dialogue manager does not invoke a general-purpose theorem prover.[3] We use a cyclical scheme for several separate reasons.

Firstly, there are dialogue moves which decompose naturally into sequences of smaller moves, some of which can also be made individually. For instance, an acknowledgement can be given explicitly (using an utterance like *Okay*), or implicitly, by making a new forward-looking utterance. It makes sense to model the implicit acknowledgement in this latter case using the same rule used to model the explicit one. To do this requires a cyclical scheme for making context updates.

Secondly, there are dialogue moves which decompose into sub-moves which have specific verbal and nonverbal reflexes. For instance, when a speaker is processing an incoming question utterance, she first of all has to recognise that it is a question. When this happens, the agent might furrow her brow, or utter a filler like *Hmm!*, and only after some time actually respond to the question. Such actions are conveniently modelled as overt side-effects of the application of cyclical context-update rules.[4]

Perhaps most importantly, the cyclical application of context-update rules has the same kind of systematicity as is found in the operation of a sentence parser or a sentence generator. In Section 1 it was argued that the benefits of a separation between declarative and procedural resources are largely due to the systematic iterative or procedural recursive algorithms which this separation permits. Our suggestion is to think of the set of utterance-processing rules as a declarative theory of 'legal dialogue context updates', and to think of the rule interpreter cycling on these rules as the dialogue equivalent of a sentence generation algorithm.

5 A simple worked example

In this section, we will give some examples of the utterance processing rules used in Te Kaitito's dialogue manager, and explain how they are used. The examples relate to the very simple dialogue given in Figure 3. After the first two utterances, the dialogue context will be the one which was shown in Figure 2.

[3] While this is of course limiting for a practical system, we feel that architectures for dialogue management can be studied relatively independently from issues to do with general-purpose reasoning techniques.

[4] Te Kaitito has in fact been used as the back-end of an animated conversational agent which performs a few simple nonverbal strategies of this kind. For details see (King et al., 2003).

We take up the story when the agent's interlocutor I

1	I	A cat was in a house
2	A	Okay
3	I	The cat barked
4	A	Okay

Figure 3: An example dialogue

makes the second assertion. After this utterance has been interpreted, the dialogue context is as shown in Figure 4. Basically, a new sentence DRS has arrived

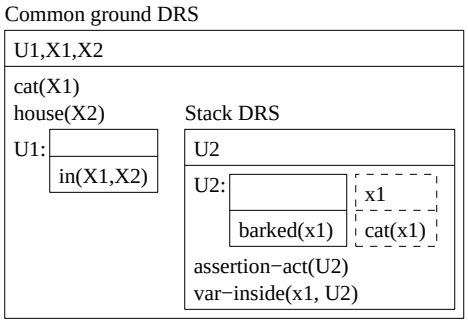

Figure 4: Context after interpretation of Utterance 3

in the stack DRS. Note that the sentence contains a presupposition DRS, which is given in dashed lines. Note also the predicates specifying the dialogue act performed by the utterance (which in this case is derived from the syntax of the sentence), and detailing which variables are mentioned in it.

Speaker-independent utterance processing rules now fire. The first rule to fire in this case is a rule which attempts to resolve the presuppositions of the utterance. (Recall that presupposition resolution is considered part of dialogue management rather than simple sentence interpretation; again see de Jager *et al.* (2002) for details.) After presupposition resolution, the context is as shown in Figure 5; the presupposition box disappears, and the variable $x1$ is bound to $X1$. In fact, no further utterance processing rules will fire in this cycle.

A speaker-specific process of response generation is now invoked. In this case, the process specifies that an assertion whose presuppositions have been fully resolved should be acknowledged, and consequently an acknowledgement is given to the speaker. A semantic representation of this acknowledgement

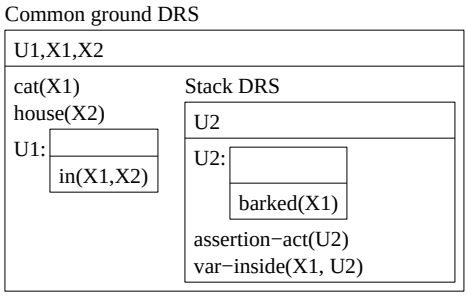

Figure 5: Context after processing of Utterance 3

sentence is added to the stack DRS, as shown in Figure 6. Notice that an acknowledgement is represented semantically as presupposing a forward-looking dialogue act. (In Te Kaitito's type hierarchy, an assertion is one such act.)

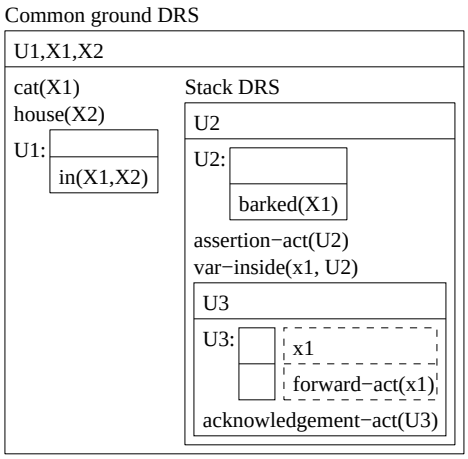

Figure 6: Context after generation of Utterance 4

The agent has now uttered the acknowledgement; it remains for it to bring its dialogue context up to date, using the same utterance processing rules as are used when processing the interlocutor's utterance. The first rule is presupposition resolution again. Crucially, the same presupposition resolution routine is invoked now as was invoked when the interlocutor's assertion was being processed. After this routine, the presupposed forward-looking dialogue act is bound to the assertion utterance (which in Te Kaitito's type hierarchy is a type of forward-looking act), the presupposition box is removed, and an explicit statement of the relationship between the assertion U2 and the acknowledgement U3 is added.

The dialogue context is now as shown in Figure 7.

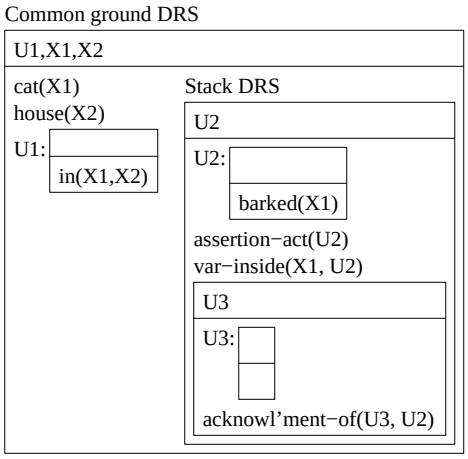

Figure 7: Context after processing Utterance 4 (1)

Finally, a second utterance-processing rule removes the acknowledgement altogether and transfers the assertion U2 to the common ground, as shown in Figure 8.

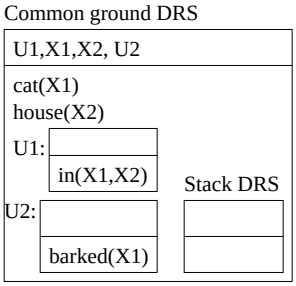

Figure 8: Context after processing Utterance 4 (2)

6 Discussion

The important thing about the example just worked through is that the utterance-processing rules which operate after each utterance is added to the stack would work in just the same way if the agent had been making an assertion and the interlocutor had been acknowledging it. The Te Kaitito system is able to generate assertions as well as interpret them, and is able to interpret acknowledgements as well as generate them. For instance, if the user asks a question, the system has to answer it with an assertive statement, and it has to be able to interpret an acknowledgement of this answer (whether the user gives this explicitly, or implicitly).

An example dialogue to illustrate this alternative pattern is given in Figure 9. During the course of

5	*I*	Which cat barked?
6	*A*	It was the blue cat.
7	*I*	Okay.

Figure 9: Dialogue with an acknowledgement by *I*

this dialogue, the utterance processing rules which update the dialogue context after the system's assertion (Utterance 6) are the same as those which update the dialogue context after the assertion by the user in Utterance 3 of Figure 3. And the utterance processing rules which update the context after the user's acknowledgement (Utterance 7) are the same as the rules which update the context after the system's acknowledgement of the user's assertion in Utterance 4 of Figure 3. The process of generating an assertion is quite different from that of interpreting one, as emphasised in Section 1, and the same goes for the processes of interpreting and generating an acknowledgement, but what this system attempts to do is to factor out the components of these two operations which are the same, and which just concern how the semantic specification of an incoming utterance can be incorporated into the current dialogue context. To the extent that this is possible, an attractive separation can be achieved between a declarative (rule-based) theory of context updates in dialogue and a procedural theory of utterance interpretation and generation.

7 Acknowledgements

This work was funded by University of Otago Research Grant MFHB10, and by the NZ Foundation for Research in Science & Technology grant UOOX02.

References

J Bos and M Gabsdil. 2000. First-order inference and the interpretation of questions and answers. In *Proceedings of Gotalog 2000. Fourth Workshop on the Semantics and Pragmatics of Dialogue.*, pages 43–50.

S de Jager, A Knott, and I Bayard. 2002. A DRT-based framework for presuppositions in dialogue management. In *Proceedings of the 6th workshop on the se-*

mantics and pragmatics of dialogue (EDILOG 2002), Edinburgh.

B J Grosz and C L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, pages 175–203.

E Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63:341–385.

H Kamp and U Reyle. 1993. *From discourse to logic*. Kluwer Academic Publishers, Dordrecht.

S King, A Knott, and B McCane. 2003. Language-driven nonverbal communication in a bilingual conversational agent. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA)*.

A Knott and N Wright. 2003. A dialogue-based knowledge authoring system for text generation. In *AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, Stanford, CA.

A Knott, I Bayard, S de Jager, and N Wright. 2002. An architecture for bilingual and bidirectional nlp. In *Proceedings of the 2nd Australasian Natural Language Processing Workshop (ANLP 2002)*.

W C Mann and S A Thompson. 1988. Rhetorical structure theory: A theory of text organization. *Text*, 8(3):243–281.

D Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA.

M Poesio and D Traum. 1998. Towards an axiomatization of dialogue acts. In J Hulstijn and A Nijholt, editors, *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues (13th Twente Workshop on Language Technology)*, pages 207–222.

D Traum, Bos J, R Cooper, S Larsson, I Lewin, C Matheson, and M Poesio. 1999. A model of dialogue moves and information state revision. TRINDI project deliverable.

R Van der Sandt. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377.