A DRT-based framework for presuppositions in dialogue management

Samson de Jager, Alistair Knott and Ian Bayard
Dept of Computer Science, University of Otago, New Zealand
{sdejager/alik/ibayard}@cs.otago.ac.nz

Abstract

In this paper we describe a dialogue system which makes use of the notion of presuppositions to provide a unifying perspective on the function of a dialogue manager. We extend van der Sandt's (1992) treatment of presuppositions in DRT to allow presuppositions to support question-answering, the generation of targeted follow-up questions, the identification of dialogue acts and the semantic disambiguation of questions.

1 Introduction

A key component of any system which engages in natural language interaction with a user is its **dialogue manager**. At a high level of abstraction, what this module does is to determine the dialogue act associated with a user's incoming utterance and to decide on a suitable dialogue act to produce in response. However, this high-level description often hides a disparate collection of specific strategies, which are closely tied to the semantics of the incoming utterance and of the response to be generated. For instance, the answer to a question relates in a very specific way to the semantics of the question, while a followup question produced in response to a partially-understood utterance relates in a very specific (but different) way to the semantics of the utterance. As a consequence of these linkages, 'dialogue manager' often ends up being a dignified word for a heterogeneous set of if-then statements.

In this paper we describe a system in which the linguistic notion of presuppositions is extended to provide a unifying perspective on the job of a dialogue manager. Te Kaitito ('the improviser/interpreter') is a web-based bilingual dialogue system, supporting the authoring and querying of a database of facts in English and Māori. The system is eventually to be embedded in an educational application, to support the learning of Māori as a second language, but at present there is no high-level 'task' underlying the dialogue between the system and the user: the user simply tells the system about states and events or asks questions about them. We are therefore mainly interested in the mechanisms that underlie 'exhaustive' answers to questions (in the sense of Ginzburg, 1996) and 'direct' answers to questions (as opposed to the indirect answers which commonly occur in task-oriented dialogue, c.f. for instance Asher and Lascarides, 1998). The system is accessible over the web, at `http://tutoko.otago.ac.nz:8080/teKaitito/` (though not every feature mentioned in this paper is fully integrated in the publically available version). An example dialogue with the system is given in Figure 1; we will use this figure to illustrate the features of the system as they are introduced. The sentence-processing mechanisms used in the system are briefly described in Section 2, and the dialogue manager is introduced in Section 3. In Section 4 we describe a range of novel ways to use the concept of presuppositions in dialogue management. In Section 5, we compare the system's model of presupposition and dialogue to a number of other existing models.

2 Syntax and semantics for Te Kaitito

Sentence interpretation and generation in Te Kaitito is performed by the LKB system

Speaker	Utterance	
User	a blue dog barked	(a)
System	Ok	
User	a red dog walked	
System	Ok	
User	which dog walked	(b)
System	the red dog it was the red dog nä te kurï whero	(c)
User	Ok	(d)
User	the blue dog chased a green cat	(e)
System	Ok	
User	a weasel barked	
System	Ok	
User	a big cat chased the weasel	
System	Ok	
User	the cat barked	
System	Ok	
User	the red dog chased the weasel	
System	Ok	
User	the blue dog walked	
System	Ok	
User	the dog ate the weasel	(f)
System	which dog ate the weasel	(g)
User	the red dog	
System	Ok	
System	Ok	
User	the blue dog chased a slow monkey	(h)
System	Ok	
User	a loud monkey ate the blue dog	
System	Ok	
User	which monkey did the blue dog chase	
System	the slow monkey it was the slow monkey nā te makimaki pōturi	
User	Ok	
User	what chased the monkey	(i)
System	the blue dog it was the blue dog nā te kurī kikorangi	

Figure 1: An extended dialogue session with Te Kaitito

(Copestake *et al.*, 2000), a grammar development environment for HPSG-like unification-based grammar formalisms. LKB is a bidirectional system: the parser and the generator both consult the same declarative grammar, so that any utterance the system can understand, it can also in principle generate. The grammar we have designed is bilingual: all rules and lexical items are associated with a LANGUAGE feature, and agreement between these features is enforced throughout a parse tree. (Any words or rules which do service in both languages can naturally remain undefined for LANGUAGE.) LKB's grammar provides a mapping between sentences and semantic representations given in a formalism called Minimal Recursion Semantics or **MRS** (Copestake *et al.*, 2000).[1] The details of MRS are not important in the present context. For more about the architecture of Te Kaitito and its motivations, see Knott *et al.* (2002).

[1]Sentences which translate or paraphrase each other are associated with the same MRS representation, which means that Te Kaitito can also function as a simple sentence translation and paraphrase system.

3 Overview of the dialogue manager

The dialogue system which embeds LKB needs to consider the role of sentences in context; we therefore decided to use DRT (Kamp and Reyle, 1993) as its main representational tool. MRS structures are converted to a modified form of DRS: a sentence's assertional content is stored in a conventional DRS, and a subsidiary DRS is created to hold any of the sentence's presuppositions, along the lines proposed by van der Sandt (1992). The DRS computed for utterance (a) in the sample dialogue is given in Figure 2.

DRSref57 e50 e51 e52 e53
h55: DOG-REL(e50 DRSref57) h55: BLUE-RELN(e50 DRSref57) h56: BARK-REL(e53 DRSref57) h56: RE-S-RELN(e51 e52 e53) h56: ASSERTION-RELN(e53)

Figure 2: DRS for *a blue dog barked*

Note that event arguments are introduced by several syntactic elements, including verbs, nouns and adjectives. The event argument introduced by a sentence's main verb also serves as the argument for a set of predicates relating to the tense and aspectual type of the sentence (for instance, RE-S-RELN encodes a particular combination of Reichenbach's speech, event and reference times), and to its syntactic mode (in our case, either assertive or interrogative).

The system's database is also stored in the form of a DRS, which is termed the **global DRS**. (We do not turn the DRS into a first-order logical representation, because the system does not currently support theorem-proving.) The user can add facts to this database by entering assertive sentences, and can query the database by asking *yes/no* or *wh*-questions. Entities in the database are termed **DRS referents**, while entities in MRSs and in sentence DRSs are termed **variables**. Figure 3 gives a snapshot of the global DRS at the point just before the first question of the sample dialogue (sentence (b)) of Figure 1) is asked.

e70 e69 e68 e67 e66 DRSref74 e61 e60 e59 e58 DRSref65
h73: RE-S-RELN(e68 e69 e70) h73: WALK-REL(e70 DRSref74) h72: DOG-REL(e67 DRSref74) h72: RED-REL(e66 DRSref74) h64: RE-S-RELN(e59 e60 e61) h64: BARK-REL(e61 DRSref65) h63: DOG-REL(e58 DRSref65)

Figure 3: Database DRS after two statements by the user

The system's dialogue model uses a hierarchical taxonomy of dialogue acts, the top levels of which are given in Figure 4. This hierarchy is, roughly speaking, a subtree of the hierarchy of dialogue acts given in Alexandersson *et al.* (1997), but it bears a resemblence to several other such taxonomies (see e.g. Allen and Core, 1997). There are two kinds of STATEMENT, both of which provide new information to the interlocutor: an ASSERTION is a new statement 'apropos of nothing', while an ANSWER is a statement which answers a question. Likewise, there are two kinds of QUERY, both of which pose a question: a QUESTION is a new query apropos of nothing, while a FOLLOW-UP QUESTION is a question about the proper interpretation of a previous utterance.

A simple model of dialogue structure is supported: a QUERY should be followed by an ANSWER; a STATEMENT of any kind should be followed by an ACKNOWLEDGE; a SENTENCE of any kind can be followed by a FOLLOW-UP QUESTION; and (at a meta-level) an UTTERANCE of any kind which doesn't follow these rules is followed by an appropriate ERROR report and ignored. FOLLOW-UP QUESTIONs allow nesting of dialogue moves in the usual way, so a familiar kind of **dialogue stack** is maintained (c.f. for instance Ginzburg's (1996) partially-ordered set of 'questions under discussion', Lemon *et al.*'s (2001) 'issues raised' stack). An example of an embedded structure is shown after utterance (f) in the sample dialogue. (Note that we currently require all ACKNOWLEDGEMENTs to be given explicitly, so at the end of a nested dialogue game, two successive ACKNOWLEDGEMENTs are given.)

An incoming utterance is interpreted by LKB, which delivers an MRS to the dialogue manager. The dialogue manager first computes the dialogue act associated with the utterance, partly from its MRS and partly from dialogue expectations (see Section 4.4 for an example of this). It also converts the MRS into a DRS, and attempts to resolve the presuppositions associated with this DRS. What happens next depends in a complex way on the success of this operation, and on the DRS, and on the dialogue act. It is at this stage that we intend a treatment of presuppositions to provide a unifying perspective.

4 Presuppositions for dialogue management

Correct resolution of presuppositional elements of a sentence requires a reasonably complex computational mechanism, even within a simple database query domain like that of Te Kaitito. We have found that this

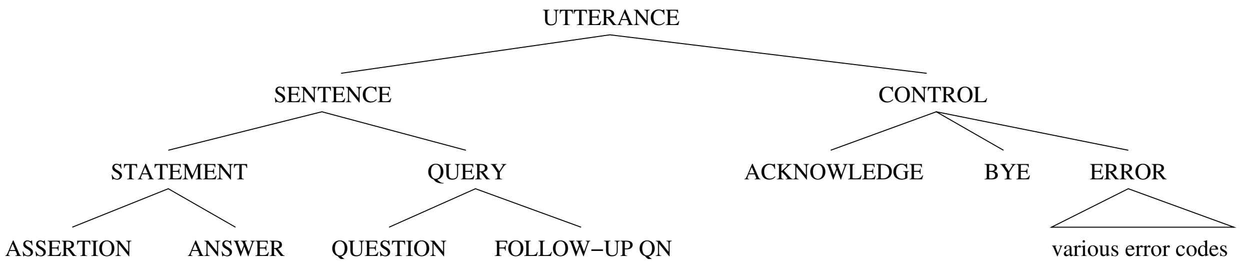

Figure 4: Dialogue act types in the dialogue model

mechanism is very similar to that required for various other tasks in dialogue management (notably question answering and classifying some dialogue acts). From a software engineering perspective it makes sense to enquire whether these similarities are close enough to allow the same software module to be used for the different tasks. This approach has been successful with only minor changes to the representation of simple presuppositions. It should be emphasised, however, that we are not claiming that all uses of this mechanism actually are instances of presupposition resolution. We only wish to draw attention to the fact that a single mechanism can be used to solve what appear to be quite distinct problems in construction of a dialogue system.

4.1 Presuppositions for computing reference

It is conventional to treat certain referring expressions (in particular definite descriptions) as presuppositional; for instance, utterance (e) (*The blue dog chased a green cat*) presupposes the existence of a uniquely identifiable blue dog. The DRS for this utterance is shown in Figure 5, with a presupposition box for *the blue dog* in the style of van der Sandt (1992). This box explicitly represents the fact that to match the variable $x4$ to a DRS referent in the global DRS we will need to find some referent for which the predicates in the box are true. Implicitly it requires uniqueness of that referent, but this is enforced by the code, not the representation. (Section 4.5 will show a case where we wish to relax this restriction.)

In cases where several DRS referents

e69 e68 e67 e66 DRSref79 x4 e70	
h73: ASSERTION-RELN(e70)	
h75: CAT-REL(e69 DRSref79)	
h75: GREEN-REL(e68 DRSref79)	
h73: RE-S-RELN(e66 e67 e70)	
h73: CHASE-REL(e70 x4 DRSref79)	
Presup-box:	e76 **x4** e77
	h78: DOG-REL(e76 x4)
	h78: BLUE-REL(e77 x4)

Figure 5: DRS for *The blue dog chased a green cat*. The marker variable for the presupposition box is shown in bold.

match the variable in a presupposition box, we use a simple representation of the saliency of referents to disambiguate. The system maintains a **saliency list**—at present, simply a list of DRS referents in order of their most recent appearance. In resolving the presuppositions of an utterance we use the saliency list to limit the number of possible distractors. The most salient individual matching the referring expression is first found. If there is another matching individual in the list within a certain **saliency range**, the presupposition is taken to be ambiguous. (The saliency range increases in relation to the depth in the list of the most salient matching individual.)

If an incoming utterance's presuppositions can be fully resolved, then if it is an assertion its DRS is merged with the global DRS, if it is a question it is answered (see Section 4.3), and if it is an answer it is acknowledged. If they cannot be resolved, a follow-up question is generated (see Section 4.2). The presupposition-handling routine thus in large part determines what happens next.

4.2 Multiple presupposition boxes for follow-up questions

Utterance (f) is a statement whose presuppositions cannot be resolved, even using the saliency list. To resolve the ambiguity the system asks a follow-up question. One innovation in our system is that when building our DRS representations we create a separate presupposition box for each presuppositional referring expression. The marker variable in the presupposition box shows which variable it is intended to disambiguate. This makes it easy to construct a specific follow-up question for the variable whose referent cannot be uniquely identified. In the example above, it is *the dog* which is ambiguous, not *the weasel*. Since these are kept in separate presup boxes, only one is queried in the follow-up question.

4.3 Presuppositions in the representation of questions and answers

Our grammar accepts a range of sentence structures specific to answers, such as *It was the dog* or *The dog*. These sentences clearly require context in order to be interpreted. There are a great number of proposals about how to model this context-dependence; Ginzburg and Sag (2000) is one particularly well worked out method. Our treatment is relatively simple: we have chosen to account for such presuppositions by modelling a question as something which introduces a meta-level predicate about the questioner's (lack of) knowledge about a domain object. For example, the DRSs for the *wh*-question *Which red cat barked* and its answer *The big cat* are shown in Figure 6. We take the *wh*-question to comprise an assertional and a presuppositional component, just like a declarative statement. Its presupposition is that some red cat $x4$ barked. Its assertional content, which is added to the discourse in the usual way, is that $x4$ is unknown—the UNBOUND-RELN predicate conveys this. Naturally, this predicate is of a different type from predicates like RED-REL, CAT-REL etc; it conveys information

```
h96: UNBOUND-RELN(e94 x4)
        e89 e90 e91 e92 e93
        x4 e94
        h95: QUESTION-RELN(e91)
P-box:  h95: RE-S-RELN(e90 e89 e91)
        h95: BARK-REL(e91 x4)
        h96: CAT-REL(e92 x4)
        h96: RED-REL(e93 x4)
```

```
x4 x11 e101 e100 e102
h107: TOKEN-IDENTITY-RELN(e102 x11 x4)
h107: RE-S-RELN(e100 e101 e102)
h107: ANSWER-RELN(e102)
        x11 e111
P-box:  h112: UNBOUND-RELN(e111 x11)
        e108 x4 e109
        h110: NEUTER-RELN(e108 x4)
P-box:  h110: SING-RELN(e108 x4)
        h110: CAT-REL(e108 x4)
        h110: BIG-REL(e109 x4)
```

Figure 6: DRS for a question (*Which red cat barked*) and its answer (*The big cat*)

about the questioner's epistemic state rather than about the world. However, this liberty allows us to deal with the presuppositions of the answer sentence in a very simple way. As the right-hand DRS in Figure 6 shows, *The big cat* contains two separate presuppositional elements, which we store in separate boxes (as discussed in Section 4.2). The first is the regular presupposition associated with the definite NP *the big cat*. The second is simply that an unknown object is present in the discourse. (For a more elaborate answer sentence such as *It was the big cat which barked* we would be able to say more about this unknown object.) The main assertional content of the answer sentence is provided by the TOKEN-IDENTITY-RELN predicate, which states that the unknown object is identical to some other variable introduced by the sentence—in our case, the variable associated with the presupposed big cat.

4.4 Follow-up questions and the stack DRS

As already mentioned, the system maintains a stack of dialogue acts to support nested follow-up questions. This raises the question of storage of the DRSs associated with the sentences associated with the dialogue acts sitting in the stack. These DRSs cannot always be merged with the global DRS, because they may (in the case of a statement producing a follow-up question) have variables that have not yet been bound to DRS referents. Our solution is to maintain an intermediary **stack DRS**, that contains a mixture of DRS referents and variables. At the point where all follow-up questions have been answered, all variables have been bound to global DRS referents (this is the function of presupposition boxes) so it is safe to merge the stack DRS into the database DRS.

The stack DRS has two other useful functions. Most straightforwardly, it provides a way of limiting the domain of 'epistemic' predicates like UNBOUND-RELN and TOKEN-IDENTITY-RELN. Occurences of the former predicate can simply be removed when the stack DRS is merged with the global DRS, and occurrences of the latter predicate can be removed by variable substitutions. A second use for the stack DRS is in determing the dialogue act associated with an incoming utterance in certain specific cases. As already mentioned, the system allows for two kinds of question, QUESTION (apropos of nothing) and FOLLOW-UP QUESTION. However, there are currently no syntactic means to distinguish between these two dialogue act types (we have not yet implemented echo questions). Fortunately, the system can use the stack DRS to disambiguate. A follow-up question can only make reference to the the entities in the utterance it is querying, or those lying higher in a chain of follow-up questions. For instance, the query *Which dog chased the cat* can be a follow-up question to the statement *The dog chased the cat* but not to the statement *The aardvark chased the cat.* We specify that an incoming question is a FOLLOW-UP QUESTION if and only if its presuppositions can be resolved using only the variables and DRS referents in the stack DRS. In a few cases this strategy will wrongly treat a QUESTION as a FOLLOW-UP QUESTION; however in these cases the answer will be the same, and the error will be overcome as soon as a question involving other entities or a statement is received.

4.5 Question answering as presupposition resolution

The use of presuppositions in representing the semantics of questions, as described in Section 4.3, also has some beneficial procedural consequences: the process of generating answers to questions occurs largely as a side-effect of the regular process of presupposition resolution. The presupposition resolution algorithm returns a list of possible lists of bindings from variables to DRS referents—these are in fact exactly what we need in order to generate the answers to questions. Multiple bindings might represent a list answer (such as *The blue cat and the red cat both chased the dog*). In the case of a yes/no question involving the indefinite article (*Did a cat chase the dog*) no distinction is made between a single binding or multiple bindings: either represents a *yes.*

Reducing the process of question-answering to the process of presupposition-resolution has the effect of simplifying the operation of the dialogue manager, which is one of the intentions in our system. However, this simplification also results in some interesting and subtle question-answering behaviour. In the interpretation of a question such as *What chased the monkey* (line (i) in the dialogue), only those monkeys which were chased will be considered as possible referents for the referring expression *the monkey.* In the example dialogue, at the point where *What chased the monkey* is asked, there are two possible referents for *the monkey.* If presupposition resolution and question-answering are treated separately, *the monkey* would therefore be interpreted as ambiguous. However in our treatment we can determine that the monkey we want is the one that was chased. This is because the DRS for the question has two presupposition boxes: one for the definite DP *the monkey* and one for the (unknown) object that chased the monkey. The same set of bindings has to satisfy both presuppositions; this means in turn that the monkey was involved in a chasing event, which removes the other monkey as a possible distractor. Presupposition resolution functions naturally here both to control the procedural component of dialogue management (the generation of an answer) and to control the semantic content of the system's responses in relation to the content of incoming utterances.

5 Comparison with existing dialogue systems and theories

This final section situates Te Kaitito in relation to some other dialogue systems and models.

5.1 Task-based dialogue

As Te Kaitito is currently being developed as a database interface, the interaction is entirely user-driven. There is no provision in the architecture for system initiative, except in asking follow-up questions. Lemon *et al.* (2001), Larsson *et al.* (2000) and Xu and Rudnicky (2000) have developed systems that drive the interaction through some form of dialogue plan on the part of the system. We hope that the approaches to dialogue we have developed with Te Kaitito will allow the addition of a module containing more complex dialogue plans, allowing the system to make *motivated* dialogue moves, rather than simply responding to user demands. This module will be essential for the successful development of the language teaching application which is the eventual aim for the project.

5.2 The dialogue stack

Lemon *et al.* (2001) suggest that the use of stacks to represent dialogue is overly restrictive, because in a task-based dialogue the system may need to shift back and forth between topics, rather than either pushing a new topic onto the stack or popping the current topic to reveal an older one. Again, the database query domain allows this simplification; however for a more general dialogue system their system of dialogue move trees may need to be adopted.

5.3 Semantics of questions and answers

We have used an extremely simple semantics for questions and answers, which is sufficient for the database applications (and perhaps for language teaching also) but does not provide a general account. Particularly, there is no provision in Te Kaitito for partial or indirect answers to questions. Asher and Lascarides (1998) provide an account which explicitly represents the status of questions as steps in a plan to gain information. In this account, a statement is considered an indirect answer to a question if it allows the questioner to complete their plan (to gain the required knowledge) only after applying some deductive or active steps. A partial answer, on the other hand, leaves the possibility of providing more information at a later date. Neither of these types of answers is necessary for the database query domain, so we have chosen not to represent them.

5.4 Public and private information states and the stack DRS

It is common in dialogue systems to distinguish between the information assumed to be shared between the participants in the dialogue (the propositions that have been stated and accepted during the dialogue) and information that has been stated but has not yet been acknowledged, so that the agent has no guarantee that the information has been understood (for example Larsson *et al.* (2000) distinguish between the 'public' and 'private' sections of the agent's information state, and include in the private section information that has been stated but not yet accepted). In Te Kaitito, this distinction corresponds quite closely to the distinction between the context DRS and the dialogue stack DRS. We believe that the use of this distinction to provide dialogue move characterisations of sentences is innovative (at least in dialogue implementations).

References

Alexandersson, J., Buschbeck-Wolf, B., Fujinami, T., Maier, E., Reithinger, N., Schmitz, B., and Siegel, M. (1997). Dialogue acts in Verbmobil-2. Technical Report 204, DFKI Saarbrúcken.

Allen, J. and Core, M. (1997). Coding dialogs with the damsl annotation scheme. In *Working notes of the AAAI Fall 1997 Symposium on Communicative Action in Humans and Machines*.

Asher, N. and Lascarides, A. (1998). Questions in dialogue. *Linguistics and Philosophy*, **23**(2), 237–309.

Copestake, A. (2000). The (new) LKB system. CSLI, Stanford University.

Ginzburg, J. (1996). Interrogatives: Questions, facts and dialogue. In S. Lappin, editor, *The handbook of contemporary semantic theory*, pages 385–422. Blackwell, Oxford.

Ginzburg, J. and Sag, I., editors (2000). *Interrogative investigations*. CSLI Publications, Stanford.

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

Knott, A., Bayard, I., de Jager, S., and Wright, N. (2002). An architecture for bilingual and bidirectional nlp. Manuscript, Dept of Computer Science, University of Otago.

Larsson, S., Ljunglof, P., Cooper, R., and Engdahl, E. (2000). GoDiS—an accommodating dialogue system. In *ANLP/NAACL 2000 Workshop on Conversational Systems*.

Lemon, O., Bracy, A., Gruenstein, A., and Peters, S. (2001). Information states in a multi-modal dialogue system for human-robot conversation. In *Proceedings of the 5th Workshop on Formal Semantics and Pragmatics of Dialogue (Bi-Dialog)*, pages 57–67.

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

Xu, W. and Rudnicky, A. (2000). Task-based dialog management using an agenda. In *ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 42–47.