GoDiS - An Accommodating Dialogue System

Staffan Larsson, Peter Ljunglöf, Robin Cooper, Elisabet Engdahl, Stina Ericsson
Department of linguistics, Göteborg University
Box 200-295, Humanisten, SE-405 30 Göteborg, Sweden
{sl,peb,cooper,engdahl,stinae}@ling.gu.se

Abstract

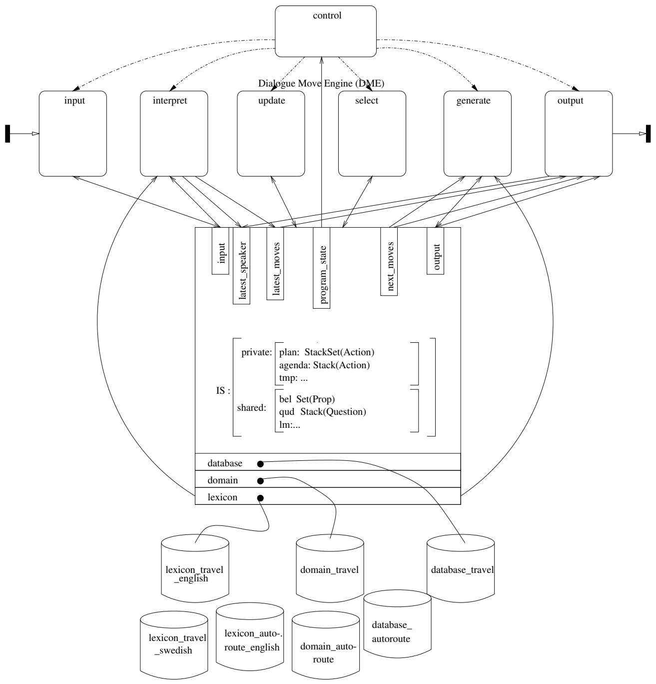
This paper accompanies a demo of the GoDiS system. Work on this system was reported at IJCAI-99 (Bohlin et al., 1999). GoDiS is a prototype dialogue system for information-seeking dialogue, capable of accommodating questions and tasks to enable the user to present information in any desired order, without explicitly naming the dialogue task. GoDiS is implemented using the TRINDIKIT software package, which enables implementation of these behaviours in a compact and natural way.

1 Introduction

This paper accompanies a demo of the GoDiS[1] system reported at IJCAI-99 (Bohlin et al., 1999). GoDiS is a prototype dialogue system for information-seeking dialogue, capable of accommodating questions and tasks to enable the user to present information in any desired order, without explicitly naming the dialogue task. GoDiS is implemented using the TRINDIKIT[2] software package developed in the TRINDI project. The TRINDIKIT is a toolkit for building and experimenting with *dialogue move engines* and *information states* (IS), We use the term *information state* to mean, roughly, the information stored internally by an agent, in this case a dialogue system. A *dialogue move engine* (DME) updates the information state on the basis of observed dialogue moves and selects appropriate moves to be performed.

2 System Description

The overall structure of the GoDiS system is illustrated below:



Like any dialogue system built using the TRINDIKIT, GoDiS consists of a number of modules, an information state, and a number of resources hooked up to the information state.

In addition to the **control** module, which wires together the other modules, there are six modules in GoDiS: **input**, which receives input[3]from the user; **interpret**, which interprets utterances as dialogue moves with some content; **generate**, which generates natural language from dialogue moves; **output**, which produces output to the user; **update**, which updates the information state based on interpreted moves; and **select**, which selects the next move(s) to perform[4]. The last two are DME modules, which means that they together make up the

[1]Work on GoDiS has been supported by the TRINDI (Task Oriented Instructional Dialogue), EC Project LE4-8314, SDS (Swedish Dialogue Systems), NUTEK/HSFR Language Technology Project F1472/1997, and INDI (Information Exchange in Dialogue), Riksbankens Jubileumsfond 1997-0134, projects.

[2]`www.ling.gu.se/research/projects/trindi/`

[3]GoDiS originally accepted written input only, but it is currently being hooked up to a speech recogniser to accept spoken input.

[4]This is done by updating the part of the information state containing the moves to be performed.

DME in GoDiS. DME modules consist of a set of *update rules* and (optionally) an update algorithm governing the order in which rules are applied. Update rules are rules for updating the information state. They consist of a rule name, a precondition list, and an effect list. The preconditions are conditions on the information state, and the effects are operations on the information state. If the preconditions of a rule are true for the information state, then the effects of that rule can be applied to the information state.

There are three resources in GoDiS: a lexicon, a database and a domain resource containing (among other things) domain-specific dialogue plans. Currently, there are GoDiS resources for a travel agency domain and the autoroute domain. Also, for each of these domains there are lexicons in both English and Swedish.

The question about what should be included in the information state is central to any theory of dialogue management. The notion of information state we are putting forward here is basically a simplified version of the dialogue game board which has been proposed by Ginzburg. We are attempting to use as simple a version as possible in order to have a more or less practical system to experiment with.

The main division in the information state is between information which is private to the agent and that which is (assumed to be) shared between the dialogue participants. What we mean by shared information here is that which has been established (i.e. grounded) during the conversation, akin to what Lewis in (Lewis, 1979) called the "conversational scoreboard". We represent information states of a dialogue participant as a record of the type shown in figure 1.

The private part of the information state includes a set of beliefs and a dialogue *plan*, i.e. is a list of dialogue actions that the agent wishes to carry out. The plan can be changed during the course of the conversation. For example, if a travel agent discovers that his customer wishes to get information about a flight he will adopt a plan to ask her where she wants to go, when she wants to go, what price class she wants and so on. The *agenda*, on the other hand, contains the short term goals or obligations that the agent has, i.e. what the agent is going to do next. For example, if the other dialogue participant raises a question, then the agent will normally put an action on the agenda to respond to the question. This action may or may not be in the agent's plan.

The private part of the IS also includes "temporary" shared information that saves the previously shared information until the latest utterance is grounded, i.e. confirmed as having been understood by the other dialogue participant[5]. In this way it is easy to retract the "optimistic" assumption that the information was understood if it should turn out that the other dialogue participant does not understand or accept it. If the agent pursues a cautious rather than an optimistic strategy then information will at first only be placed in the "temporary" slot until it has been acknowledged by the other dialogue participant whereupon it can be moved to the appropriate shared field.

The (supposedly) shared part of the IS consists of three subparts. One is a set of propositions which the agent assumes for the sake of the conversation and which are established during the dialogue. The second is a stack of questions under discussion (QUD). These are questions that have been raised and are currently under discussion in the dialogue. The third contains information about the latest utterance (speaker, moves and integration status).

3 Accommodation in GoDiS

Dialogue participants can address questions that have not been explicitly raised in the dialogue. However, it is important that a question be available to the agent who is to interpret it because the utterance may be elliptical. Here is an example from a travel agency dialogue[6]:

```
$J: what month do you want to go
$P: well around 3rd 4th april / some time
there
$P: as cheap as possible
```

The strategy we adopt for interpreting elliptical utterances is to think of them as short answers (in the sense of Ginzburg (Ginzburg, 1998)) to questions on QUD. A suitable question here is *What kind of price does P want for the ticket?*. This question is not under discussion at the point when P says "as cheap as possible". But it can be figured out since J knows that this is a relevant question. In fact it will be a question which J has as an action in his plan to raise. On our analysis it is this fact which enables A to interpret the ellipsis. He finds the matching question on his plan, accommodates by placing it on QUD and then continues with the integration of the information expressed by *as cheap as possible* as normal. Note that if such a question is not available then the ellipsis cannot be interpreted as in the dialogue below.

A. What time are you coming to pick up Maria?
B. Around 6 p.m. As cheap as possible.

[5]In discussing grounding we will assume that there is just one other dialogue participant.

[6]This dialogue has been collected by the University of Lund as part of the SDS project. We quote a translation of the transcription done in Göteborg as part of the same project.

$$
\text{IS}: \left[\begin{array}{lcl}
\textsc{private} & : & \left[\begin{array}{lcl}
\textsc{plan} & : & \textsc{StackSet(Action)} \\
\textsc{agenda} & : & \textsc{Stack(Action)} \\
\textsc{bel} & : & \textsc{Set(Prop)} \\
\textsc{tmp} & : & \left[\begin{array}{lcl}
\textsc{bel} & : & \textsc{Set(Prop)} \\
\textsc{qud} & : & \textsc{Stack(Question)} \\
\textsc{lu} & : & \left[\begin{array}{lcl}
\textsc{speaker} & : & \textsc{Participant} \\
\textsc{moves} & : & \textsc{assocSet(Move,Bool)}
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\textsc{shared} & : & \left[\begin{array}{lcl}
\textsc{bel} & : & \textsc{Set(Prop)} \\
\textsc{qud} & : & \textsc{StackSet(Question)} \\
\textsc{lu} & : & \left[\begin{array}{lcl}
\textsc{speaker} & : & \textsc{Participant} \\
\textsc{moves} & : & \textsc{assocSet(Move,Bool)}
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 1: The type of information state we are assuming

This dialogue is incoherent if what is being discussed is when the child Maria is going to be picked up from her friend's house (at least under standard dialogue plans that we might have for such a conversation).

Question accommodation has been implemented in GoDiS using a single information state update rule **accommodateQuestion**, seen below. When interpreting the latest utterance by the other participant, the system makes the assumption that it was a **reply** move with content A. This assumption requires accommodating some question Q such that A is a relevant answer to Q. The check operator "answer-to(A, Q)" is true if A is a relevant answer to Q given the current information state, according to a (domain-dependent) definition of question-answer relevance.

$$
\begin{array}{ll}
\textsc{rule:} & \textbf{accommodateQuestion} \\
\textsc{class:} & \textbf{accommodate} \\
\textsc{pre:} & \left\{\begin{array}{l}
\text{val(} \textsc{shared.lu.speaker}\text{, usr)} \\
\text{in(} \textsc{shared.lu.moves}\text{, } \textbf{answer}(A) \text{)} \\
\text{not (lexicon :: } \textbf{yn\_answer}(A) \text{)} \\
\text{assoc(} \textsc{shared.lu.moves}\text{, } \textbf{answer}(A)\text{, false)} \\
\text{in(} \textsc{private.plan}\text{, raise}(Q) \text{)} \\
\text{domain :: relevant\_answer}(Q,\ A)
\end{array}\right. \\
\textsc{eff:} & \left\{\begin{array}{l}
\text{del(} \textsc{private.plan}\text{, } \textbf{raise}(Q) \text{)} \\
\text{push(} \textsc{shared.qud}\text{, } Q \text{)}
\end{array}\right.
\end{array}
$$

After an initial exchange for establishing contact the first thing that P says to the travel agent in our dialogue is "flights to paris". This is again an ellipsis which on our analysis has to be interpreted as the answer to a question (two questions, actually) in order to be understandable and relevant. As no questions have been raised yet in the dialogue (apart from whether the participants have each other's attention) the travel agent cannot find the appropriate question on his plan. Furthermore, as this is the first indication of what the customer wants, the travel agent cannot have a plan with detailed questions. We assume that the travel agent has various plan types in his domain knowledge determining what kind of conversations he is able to have. Each plan is associated with a task. E.g. he is able to book trips by various modes of travel, he is able to handle complaints, book hotels, rental cars etc. What he needs to do is take the customer's utterance and try to match it against questions in his plan types in his domain knowledge. When he finds a suitable match he will accommodate the corresponding task, thereby providing a plan to ask relevant question for flights, e.g. when to travel?, what date? etc. Once he has accommodated this task and retrieved the plan he can proceed as in the previous example. That is, he can accommodate the QUD with the relevant question and proceed with the interpretation of ellipsis in the normal fashion.

This example is interesting for a couple of reasons. It provides us with an example of "recursive" accommodation. The QUD needs to be accommodated, but in order to do this the dialogue task needs to be accommodated and the plan retrieved. The other interesting aspect of this is that accommodating the dialogue task in this way actually serves to drive the dialogue forward. That is, the mechanism by which the agent interprets this ellipsis, gives him a plan for a substantial part of the rest of the dialogue. This is a way of capturing the intuition that saying *flights to Paris* to a travel agent immediately makes a number of questions become relevant.

Task accommodation and retrieval of the relevant plan has been implemented in GoDiS by two information state update rules.

4 Sample dialogues

We will give some examples of GoDiS dialogues in the English travel agency domain. One of the features of the TRINDIKIT is that it provides methods for inspecting the information state at any stage of a dialogue, and also to see which rules are applied. Unfortunately, because of space limitations, we are not able to include examples of this in this paper; however, it is an integral part of our demo.

The first dialogue shows the typical behaviour of dialogue systems for information-seeking: each question is answered in turn, and the user always gives exactly the information requested.

```
$S> Welcome to the travel agency!
$U> hello
```

```
$S> What can I do for you?
$U> I want price information on a trip
please
$S> How do you want to travel?
$U> by plane
$S> What city do you want to go to?
$U> gothenburg
$S> What city do you want to go from?
$U> london
$S> Do you want a return ticket?
$U> yes
$S> What month do you want to leave?
$U> april
$S> What class did you have in mind?
$U> as cheap as possible
$S> It will cost 50 crowns
$U> thanks, bye!
$S> Thank you for your visit!
```

In contrast, the second dialogue (below) shows how GoDiS is able to accommodate the task and integrate information in the order that the user gives it.

```
 $S> Welcome to the travel agency!
$U> flights to paris
$S> What city do you want to go from?
$U> gothenburg, in april, as cheap as
possible
$S> Do you want a return ticket?
$U> no
$S> It will cost 50 crowns
$U> thanks, bye!
$S> Thank you for your visit!
```

5 Limitations

GoDiS is a small-scale prototype and as such it suffers from the familiar drawbacks of many experimental systems: its lexicons and databases are very small, and the domain knowledge is limited. The dialogue plans cover only a small (but important) part of the kind of dialogues that can take place in the travel agency and autoroute domains. We hope to be able to extend both the number of domains and the scale of each domain in the future.

Also, the focus in the development of GoDiS has been on the dialogue management, i.e. the dialogue move engine and the information state. Less effort has been put into the interpretation and generation modules, which currently use (a slightly extended version of) keyword spotting and partially canned text, respectively. We hope to replace these by plugging in existing interpretation and generation technology to GoDiS.

For the tasks that GoDiS currently is able to handle, full-blown semantics is not needed. We use a very limited semantics where propositions are essentially feature-value pairs. As a consequence, GoDiS is e.g. not able to handle dialogue with more than one referent; for this, the information state would have to be amended with a set of referents, and propositions would have to include referent information. This is an area where we hope to improve GoDiS in the near future.

Speech recognition and synthesis is currently being added to GoDiS, but at the time of writing only written input and output is available.

6 Contributions

Currently, the main contribution of GoDiS is perhaps to show how an extended notion of accommodation can serve to make dialogue systems easier to interact with, by letting the user decide how and in what order to present information to the system. Also, the fact that accommodation can be implemented simply by adding three update rules indicates that information state update rules provide a natural and compact way of implementing dialogue strategies. An important issue for future research is the relation of question and task accommodation to plan recognition approaches to dialogue (Sidner, 1985).

GoDiS also features a simple grounding strategy which is nevertheless sufficient in many cases. The grounding mechanism is implemented by three update rules. It is possible to switch resources in mid-dialogue, e.g. to change language. Also, GoDiS is easily reconfigurable to new information-seeking domains. To adapt GoDiS to a new domain, one needs to supply a database, a lexicon and domain knowledge, including a set of dialogue plans. The GoDiS modules or information state don't need to be changed in any way.

In general, as an example of a dialogue system implemented using the TRINDIKIT package, GoDiS shows how the information state approach is useful for clarifying and comparing theories of dialogue, and for exploring new solutions.

References

P. Bohlin, R. Cooper, E. Engdahl, and S. Larsson. 1999. Information states and dialogue move engines. In J. Alexandersson, editor, *IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

J. Ginzburg. 1998. Clarifying utterances. In J. Hulstijn and A. Niholt, editors, *Proc. of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 11–30, Enschede. Universiteit Twente, Faculteit Informatica.

D. K. Lewis. 1979. Scorekeeping in a language game. *Journal of Philosophical Logic*, 8:339–359.

C. L. Sidner. 1985. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1):1–10, February.