UNITS OF DIALOGUE MANAGEMENT: AN EXAMPLE

Paul Heisterkamp
Daimler-Benz AG
Research and Technology
D-89013 Ulm, Germany
heisterkamp@dbag.ulm.DaimlerBenz.COM

Scott McGlashan
Swedish Institute of Computer Science
Box 1263
S-16428 Kista, Sweden
scott@sics.se

ABSTRACT

This paper concerns dialogue management for spoken dialogue. We show why we do not use speech-act related units or intentions. We base our approach on belief states of the system. Layered units are used to construct a pragmatic interpretation of these states and to determine the dialogue continuation as a local optimisation over a set of dynamic dialogue goals. We point to systems that successfully employ this approach.

1. INTRODUCTION

Recently, Cohen /4/ gave an overview of the state of the art in dialogue modelling. He distinguished between approaches based on the notion of dialogue grammars, those based on plans and intentions, and a third approach, newly emerging, that regards dialogue as a joint activity. We see our approach as belonging to this latter area. Our explicit research goal is, in Cohen's words, 'to develop algorithms and procedures to support a computer's participation in a co-operative dialogue'. Our approach is based on a layered set of units that, taken together, model the dialogue as a combination of belief and intention states of the system (cf. /3/). In the following, we will first outline why we do not explicitly use the notion of the user intention as a basis for dialogue structure. We then explain the use of several layers of units, their scope and the way they are mapped onto each other. Next, we show how these units are used not to model a dialogue as an overall whole, but to determine the local continuation of the dialogue by local optimisation. We finally point to some application examples.

2. INTENTION AND INTERPRETATION

Both grammar and plan based dialogue models are dependent on assigning an intention function to an utterance. In case of the user's utterances, this assignment mostly employs a notion derived from that of speech acts. User utterances are analysed linguistically, and the resulting description is then categorised as belonging to, or realising, one or more dialogue acts. The inventory of these dialogue acts is set up through the analysis of a corpus of examples. Dialogue structure is described in terms of these acts, which serve as the basic unit. The corpus yields a number of sequences of units, which, in the grammar approach are used to set up a finite state grammar describing the possible and therefore legal dialogues. Dialogue management then consists of finding an overall optimal path through this complex act-transition network and taking the next system act as the dialogue continuation (cf. e.g. /1/). In the plan based paradigm, the acts are seen as indicating a plan on the part of the user, the dialogue continuation being determined by finding a complementary plan on the part of the system and performing it's next step.

It is a fundamental prerequisite for both approaches to dialogue management that the external events are correctly classified, i.e. that the user utterances are assigned the appropriate speech acts. This assignment task is usually seen as an interpretation: the user's intention of the act s/he wanted to perform when making that specific utterance. The idea is that this intention can be seen somehow as a separable operator over the semantic content, be it co-determined by the context or not. It is, at least, very difficult to infer such an intention, which is basically a mental state, from such scarce evidence as an acoustic signal, and it presupposes that recognition and understanding are near to perfect and, moreover, that humans do not make mistakes (cf. /6/, /9/). Such an interpretation may be possible in an ex-post analysis of a dialogue as a whole, where consequences can be taken into account, but in an ongoing dialogue it can, at most, be an inspired guess. Furthermore, the notion of speech acts or intentions leads to several theory-induced problems in dialogue management, e.g. forced assignment of exactly one (unary) operator to an utterance, 'incomplete' dialogue in terms of grammar, ‘indirect’ acts, etc.

3. LAYERED UNITS

We do not make use of the notion of acts as far as the dialogue partner, i.e. the user, is concerned. Rather, the continuation of the dialogue is based on the results of the contextual semantic interpretation of the utterance (cf. /8/), with all its possible deficiencies, and on monitoring changes in the belief state of the system. The units used at this level do not permit a direct computation of the continuation. In order to arrive at units which do, there are two more levels of units. The system performs two mapping steps from the belief state level to the dialogue level.

3.1 Contextual Functions

The first level of units is that of contextual functions. In the contextual interpretation (cf. /6/, /8/) the belief state of the system undergoes one or more changes. The contextual functions of an utterance describe the type of change that the interpretation brought about. There can be more than one function to an utterance. Every semantic item recognised as relevant for the dialogue receives a function assignment. Thus, the functions are parameterised to semantic items. Currently, as shown in Figure 1, only items that are relevant to the task are taken into

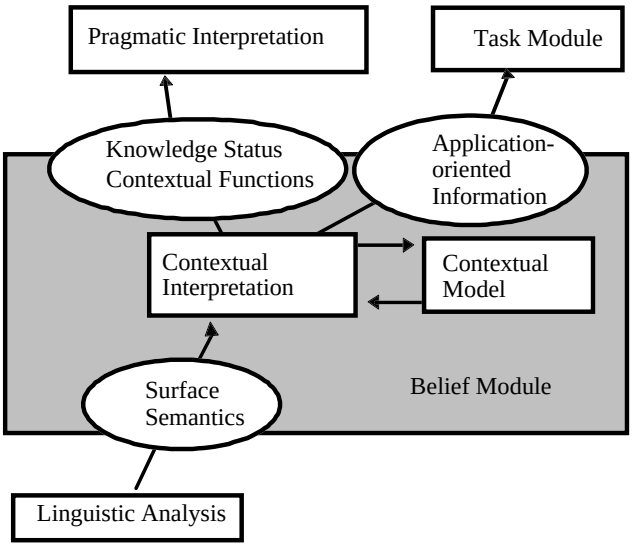

Figure 1: Schema of the Contextual Interpretation

account, but in principle, any other semantic item could also be treated in the same way. So, the approach we illustrate here is not limited to 'form filling' dialogues (cf. /4/). For example, in the train timetable application the user utterance "Munich at 8 o'clock" leads to the following surface-semantic description:

```
[thecard:2,
 1:[semantics:[id:D,
        type:location,
        thecity:[id:E, type:city, value:muenchen],
        modus:[ref:nonpro]]],
 2:[semantics:[id:H,
        type:time,
                thehourpoint:[id:I,
                type:hour_point,
                thehour:[id:J, type:hour, value:8]],
                        st_blob:at, modus:[def:a]]]].
```

Interpreted in the context of a request from the system for a destination and departure time, this structure yields a task level interpretation of two items, viz. the goal city and the departure time. Both these items are new to the system, i. e. the contextual model did not contain instances of these before. So, the contextual functions of both semantic items are described by the unit 'new_for_system (goalcity:munich)' and 'new_for_system (source-time: [0800, 2000])'. In cases of underspecification, like "8 o'clock", the contextual interpretation can forward a list of possible interpretations. So far, we have not found it necessary to multiply the entities at this level beyond the five following:

- new_for_system(X).
- repeated_by_user(X).
- inferred_by_system(X).
- modified_by_user(X).
- negated_by_user(X).

The determining factor for the choice of the units is whether some type of change in the contextual model has an influence on the system continuation or not. These functions are not meta-semantic in the sense that they are derived from the semantics themselves, rather they are pragmatic. Likewise, utterances, or parts thereof, that are marked as pragmatic (or as dialogue markers) already on the surface level, such as 'yes' and 'no', are given over to the pragmatic interpretation immediately.

3.2 Dialogue Goals

The contextual functions in turn are interpreted pragmatically. This interpretation takes into account the pragmatic context or dialogue state. A dialogue state is expressed as a set of dialogue goals that also refer to semantic objects. This is the second level of units. The functions are evaluated as to whether they solve a goal, modify it or introduce a new one. Communication with the application system determines whether the task itself wants to introduce a new semantics object and thus a new dialogue goal, or whether the task has been completed.

The dialogue functions are mapped onto the dialogue goals in the following way. The function 'new_for_system(X)' introduces a goal 'confirm(X)' or 'specify([X,Y])', if '[X,Y]' is a list of possible values as the result the interpretation of an underspecified expression. This goal is checked against the existing goal set, so as to determine whether there is already a goal pertaining to the semantic item X. If there is not, then the 'confirm(X)' or 'specify([X,Y])' goal is simply added to the set. If the semantic item is already present in the goal set, a set of rules is used to determine the way in which goals modify each other. For example, with 'confirm' or 'specify' goals, existing 'request' goals (cf. below) are simply cancelled. A counter keeps track of how many times this item was addressed in the course of the dialogue.

The function 'repeated_by_user(X)' will cancel any existing 'confirm' goals for semantic item X. If there is no goal contradicting this, the pragmatic interpretation will also inform the Belief Module that item X is known with a high degree of certainty, so that a modification of the item should be made more difficult. The same effect can be triggered by the threshold counter value. Thus, the system uses a mechanism similar to that of 'discourse pegs' (cf. /10/).

The function 'inferred_by_system(X)' is treated as if introducing a 'confirm(X)' goal, but with a difference in the determination of the dialogue continuation (cf. below).

The function 'modified_by_user(X)' indicates that, for any reason whatsoever, there is something wrong with the system's interpretation of the item X. This pragmatic interpretation has two effects. First, reacting to the evidence that there has been some difficulty, it will trigger the dialogue meta-strategy to select the next lowest level of transaction (cf. below). Second, it will both remove any other goal pertaining to item X, and introduce a

'repair_confirm (X)' goal to the goal set. This goal will be used in selecting a specific formulation of the confirmation request and, together with the counter value for item X and its 'discourse peg' state, will also force the recogniser and the parser to operate in a mode which strongly prefers an analysis compatible to the semantic dialogue predictions (cf. /7/, /13/).

Similarly, the function 'negated_by_user(X)' will move the current strategy to 'repair(X)', remove any goal pertaining to item X, replacing it with a 'repair_request' goal. This may force both parser and recogniser down to a mode where they only accept input that fits the dialogue predictions ('rigid' mode).

Dialogue markers, such as 'yes', 'no', 'pardon' etc. operate directly over the current goal set. For confirm goals that are currently being realised, a 'yes' confirms them all, a 'no' resets all these goals to an open request, a 'pardon'-type utterance will reinstate all currently active goals, leading to a (possibly reformulated, cf. /15/) repetition of the previous system utterance.

Dialogue goals are also introduced by other modules in the system. The application interface introduces task goals, such as a request for information required by the application system. The dialogue management may itself introduce goals which have phatic or meta-dialogic function, like the greeting and self-identification at the beginning of the dialogue, a good bye at the end, or notification of misunderstanding in case of recogniser or parser failure. We currently utilise twelve types of goals.

3.3 IRE-Structure and Dialogue Strategy

The dialogue goals form a conflict set. A dialogue strategy operates over this set in order to determine the best dialogue continuation. For this, the goals are declaratively grouped as to belong to one of three classes (or units), viz. initiative, reaction and evaluation (IRE) (cf. /2/, /14/). A request goal, for example, is an initiative, a confirm goal is an evaluation, etc. The IRE schema describes dialogues at an abstract level as recursively consisting of an initiative, followed by a reaction, followed by an (optional) evaluation, which again may be another IRE schema beginning with an initiative. We do not use the recursion, but implement the schema sequentially, as the dialogue coherence is dependent on the semantic items being negotiated.

The dialogue strategy determines how initiatives, reactions and evaluation are ranked and combined. In general, evaluations that are not initiatives are ranked higher than reactions, and reactions have precedence over initiatives. This ensures that answers to question are realised earlier than questions by the system, thus generating adjacency pair sequences.

The combination of goals that may be realised in one system utterance is dependent on the state of the dialogue. The standard setting is that any number of reactions (except confirm goals for inferred items) and one initiative may be realised at the same time, making the dialogue fast. A meta-strategy of degradation and recovery may move this setting to single reactions and single initiatives as shown in Figure 2. The meta-strategy operates over counters attached to semantic items, so that re-negotiation of an item, while incrementing its counter, will also indicate that there has been some problem with this item and that the strategy should degrade to make recognition and interpretation less prone to errors. Leading reactions combine a reaction with a pseudo-initiative, e. g. “Please answer yes or no!”, but they remain reactions all the same. How they are realised is solely dependent upon the current dialogue strategy. The meta-strategy mechanism is responsible for the repair behaviour of the system. Because it is on a very abstract level, and completely within the framework of the IRE schema, repair does not have to be modelled as a special case in the dialogue.

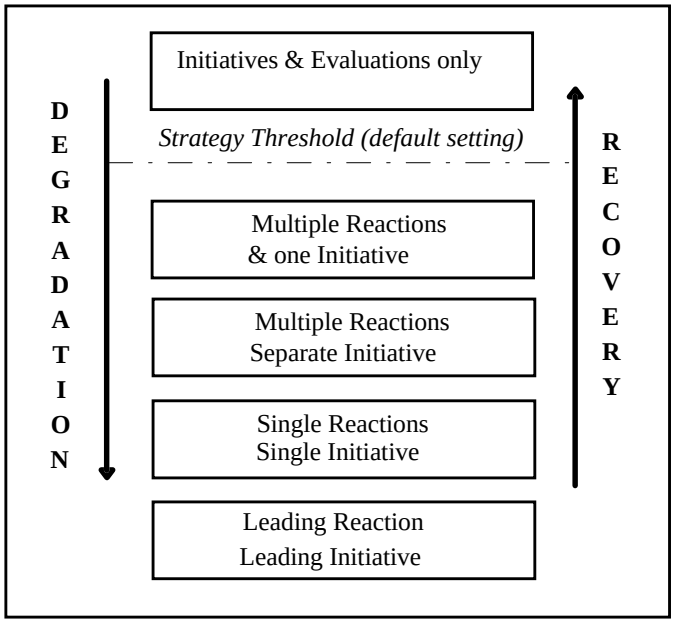

Figure 2: Dialogue strategies and meta-strategy, cf. /9/.

4. SYSTEM DIALOGUE ACTS

Turning to the system’s utterances, it is clear from the goals of the system what kind of dialogue continuation it wants to elicit from the user. Therefore, at this level, we can talk about acts proper. As these acts are also parameterised by semantic items (or operate over them), and as more than one semantic item can be realised in one system utterance, which could also consist of more than one sentence, we avoid the term 'speech act' here, and prefer 'dialogue acts' instead.

The dialogue acts of the system are realised first by determining the appropriate semantic structures for the semantic items that are the contents of the dialogue goals. These structures, which are rather rudimentary in the current implementations, are augmented with the sentence type information in a message planning module, which in turn submits these structures to a generator.

5. APPLICATIONS

We have shown how we use layered units on different levels of abstraction to manage spoken dialogue. Two of the main

advantages of this approach are a) that it does without notions of intention on the dialogue partners side, and b) that it describes dialogues in a very generic manner. This genericity is due to the degree of abstraction reached through the layering of units. On the level of dialogue continuation, a simple schema makes it possible to obtain very complex behaviour through the combination of dialogue goals pertaining to semantic items. The determination of the continuation does not have to take into account possible continuations or goals pertaining to overall 'plans', nor a notion of well-formedness that goes along with a dialogue grammar. The most important thing is to find the best local continuation in terms of a system utterance, based on the actual dialogue situation. Or, in Tom Wachtel's words: "We view dialogue as less like a chess match and more like a game of tennis. The next move is far more crucial than the grand strategy." (/16/).

To adapt the system to a new information service, it is not necessary to model specific dialogues. Rather, modelling the tasks and the application system (and, of course, the discourse world and language coverage) is sufficient. Our dialogue management system is currently running in several such applications with continuous speech input without any modifications to the dialogue structure level being necessary. There are information-providing applications like train timetable and flight enquiry systems, but also information seeking applications, like a road map update for long term and short term modifications, used for regional traffic management in the Stuttgart area. We are currently working on telephone-based applications for direct insurance and call management.

The dialogue management model has also been extended for multimodal applications where a speech interface is integrated into a direct manipulation environment (cf. /11/). Interpretation of graphical input is based on the same semantic and pragmatic structures required for spoken language, although the structures are less complex to process due to the absence of underspecified input like anaphora and ellipsis. The algorithm for realising system dialogue acts has been modified to allow generation of graphical output, and the semantic and pragmatic interpretative functions have been enhanced to handle 'command and control' utterances. However, the basic principles of our dialogue management model used in speech-only applications remain intact. This extended model is now being integrated with virtual reality applications to allow users to navigate and manipulate objects in a virtual world by means of speech input (cf. /12/).

6. REFERENCES

1. Bateman, J., Hagen, E., Stein, A. (1995): Dialogue modelling for speech generation in multimodal information systems, in: /5/.
2. Bilange, E. (1992): *Dialogue personne-machine*. Paris.
3. Bunt, H. (1989): Information dialogues as communicative action in relation to partner modelling and information processing, in: Taylor, M. M. et al. (eds.): *The Structure of Multimodal Dialogue*. Amsterdam: North Holland.
4. Cohen, P. (1996): Dialogue Modelling, in: Cole, R. A. et al. (eds.) (1996): *Survey of the State of the Art in Human Language Technology*. Oregon Graduate Institute of Technology, http://www.cse.ogi.edu/ CSLU/ HLTsurvey/
5. Dalsgaard, P. et al. (eds.) (1995): *Proc. of the ESCA Tutorial and Research Workshop on Spoken Dialogue Systems - Theories and Applications*. Vigsø, Denmark, May 30 - June 2, 1995
6. Giachin, E., and McGlashan, S. (1996): Spoken Language Dialogue Systems, in: Church, K., Young, S., and Bloothooft, G. (eds) *Corpus-based Methods in Language and Speech Processing*. Amsterdam: Kluwer.
7. Hanrieder, G. (1996): *Inkrementelles Parsing gesprochener Sprache mit einer linksassoziativen Unifikationsgrammatik*. PhD Thesis, Univ. Erlangen-Nürnberg.
8. Heisterkamp, P., McGlashan, S., and Youd, N. (1992): Dialogue semantics for an oral dialogue system, in: *Proceedings of ICSLP-92*, Banff.
9. Heisterkamp, P. (1993): Ambiguity and uncertainty in spoken dialogue, in: *Proceedings of Eurospeech 1993*, Berlin.
10. LuperFoy, S. (1995): Implementing file change semantics for spoken language dialogue managers, in: /5/.
11. McGlashan, S. (1996): Multimodal Dialogue Management, in: *Dialogue Management in Natural Language Systems*. Proc. of the 11th Twente Workshop on Language Technology, Enschede, The Netherlands.
12. McGlashan, S., and Axling, T. (1996): Talking to Agents in Virtual Worlds. In: *Proc of 3rd UK VR-SIG Conference*, Leicester, England.
13. Mecklenburg, K.; Hanrieder, G.; Heisterkamp, P. (1995): A Robust parser for continuous spoken language using PROLOG. In: *Proc of Natural Language Understanding and Logic Programming 1995*, Lisbon, Portugal.
14. Moeschler, J. (1989): *Modélisation du dialogue*. Paris.
15. Poller, P., Heisterkamp, P. (1995): *EFFENDI - Effizientes Formulieren von Dialogbeiträgen.* Daimler-Benz Technischer Bericht F3-95-014 (Internal report)
16. Wachtel, T. (assumed authorship) (1995): Robust Understanding. http://www.cre.canon.co.uk/robust.html