Gefördert vom
bmb+f
Bundesministerium für
Bildung, Wissenschaft,
Forschung und Technologie

Verbmobil
Verbundvorhaben

Some Ideas for the Automatic Acquisition of Dialogue Structure

Jan Alexandersson

DFKI GmbH

Vm Report 157
Juni 1996

Juni 1996

Jan Alexandersson

DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany

Tel.: (0681) 302 - 5347
Fax: (0681) 302 - 5341
e-mail: `janal@dfki.un-sb.de`

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 01IV101K/1 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei dem Autor.

Some Ideas for the Automatic Acquisition of Dialogue Structure

Jan Alexandersson *
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken
Germany
janal@dfki.uni-sb.de

Abstract

We are reporting on some initial results on the automatic acquisition of plan operators for a plan recognizer. The operators are derived from the VERBMOBIL corpus of negotiation dialogues hand-annotated with dialogue acts. The corpus is pre-classified and a set of plan operators is derived for every class. The plan operators are then tested on a set of unseen data. We also show some initial results.

1 Introduction

The VERBMOBIL project [13, 5] is a long term project founded by the German ministry for Education, Science, Research, and Technology, for developing an automatic interpreting system for taks oriented face-to-face dialogues. The main work so far has been concerned with the translation of natural spoken German into English, in the domain of appointement scheduling.

The project involves 31 industrial and academic partners concerned with a broad range of activities spanning from data collection, speech recognition, machine translation to pure system integration. The current status of the working system consists, from the dialogue module's point of view, of two types of processing approaches – deep and shallow. Both lines use the output from the speech recognition components (speech recognition and prosody) as input. The deep processing line uses a more traditional linguistic approach consisting of a syntax-semantic parser, a transfer module, and a tactical generator. The shallow line uses message extraction techniques to analyse, transfer, and generate. The semantic evaluation component,

*This work was funded by the German Federal Ministry for Research and Technology (BMBF) in the framework of the VERBMOBIL Project under Grant 01IV101K/1. The responsibility for the contents of this study lies with the authors. The author wishes to thank Elisabeth Maier and Norbert Reithinger for comments on earlier drafts of this paper, and Ralf Engel for his help with the implementation.

This paper is a reprint from the Proceedings of the Eleventh Twente Workshop on Language Technology, TWLT 11, Dialogue Management in Natural Language Systems, June 19-21, 1996

and the dialogue component are concerned with providing contextual information and resolving transfer relevant ambiguities.

The dialogue component of the VERBMOBIL system [1] consists of 3 main components.

The statistical component providing different modules in the system with top-down predictions.

The dialogue memory consisting of three structures, the dialogue sequence memory, the thematical structure, and the intentional structure.

The plan recognizer constructing, for instance, the intentional structure.

The dialogue processing is centered around dialogue acts [4], which are used as basic entities for both the prediction process as well as the dialogue sequence memory, and the plan recognizer.

This paper describes some ideas how one can utilize a hand-annotated corpus for automatic derivation of plan operators for a plan recognizer. We show how these ideas enable us to overcome some problems we experienced when we tried to hand code the operators. The paper is structured as follows. In section 2 the intentional structure is presented. A short introduction to our dialogue act hierarchy, and a survey of the plan recognizer is given. Section 3 gives some hints on what problems we are faced with when we want to build the structure. We show how the corpus was used in section 4. We present some of the results in section 5, and conclude the paper in section 6.

2 The Intentional Structure

From our perspective, dialogue acts represent the intended meaning of an utterance and abstract over the possible linguistic realizations. We have defined a set of 42 domain dependent as well as domain independent dialogue acts [4]. Some of them are purely illocutionary, e.g. *requesting a proposal* for a date or *requesting a comment* on a proposal, but they can also comprise propositional content, e.g. *proposing a date* or *giving a reason* for rejecting a proposed date. Traditional *illocutionary acts* as they were proposed by Austin [3] and later integrated into Searle's theory of speech acts [9] aim at a rather coarse-grained typology detached from their propositional content.

The intentional structure is a tree-like structure mirroring different abstract levels of the intentions of the dialogue. It is built of four levels (see fig 2).

The Dialogue Act Level implements, with some minor extensions, the dialogue act hierarchy.

The Turn Level connects the utterances inside a turn.

The Phase Level distinguishes the three dialogue phases greeting phase, negotiation phase and closing phase.

The Dialogue Level spans over the whole dialogue, eventually distinguishing more negotiations.

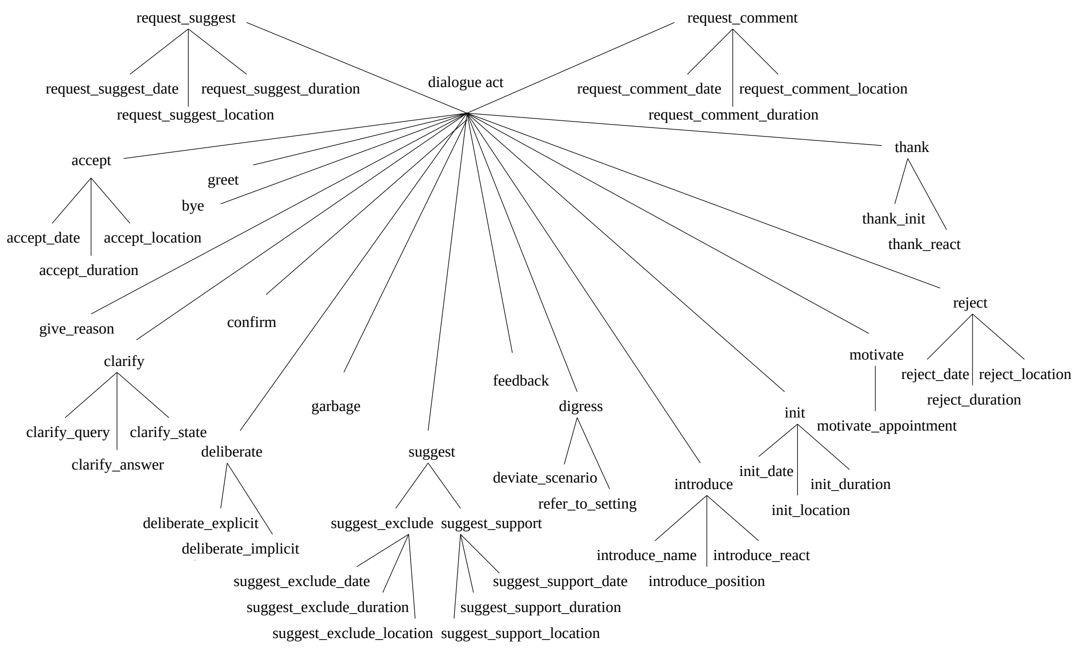

Figure 1: The dialogue act hierarchy

The reasons for dividing the structure are.

- The dialogue act hierarchy is used as a way of abstraction from the actual topic – it should not matter whether we negotiate a time or a place.
- We are interested in *reduction*, i.e. condensing the turn to its important, information carrying parts. The turns level is also a convenient way of making the plan recognizer robust against *gaps* (see section 3) in the input.
- The phases must be computed because of translation purposes – translation ambiguities can be resolved with dialogue phase information.

The turn level is the most problematic layer, and the aim of this work is to explore some ideas about how one can derive the plan operators describing the structures of turns automatically from an annotated corpus.

Building the Intentional Structure

There are a lot of similarities between plan recognition and parsing as pointed out by Vilain [12]. There are many advantages to choosing a parsing approach since the area is quite explored and there are a lot of efficient algorithms described in the literature. We have chosen a top down left-to-right parsing algorithm with destructive backtracking to build the intentional structure. There are several reasons for that. Since we know that the dialogue is about negotiaton, we make use of this knowledge. The plan operators also maintain an internal context which is “carried around” in the tree as it is constructed.

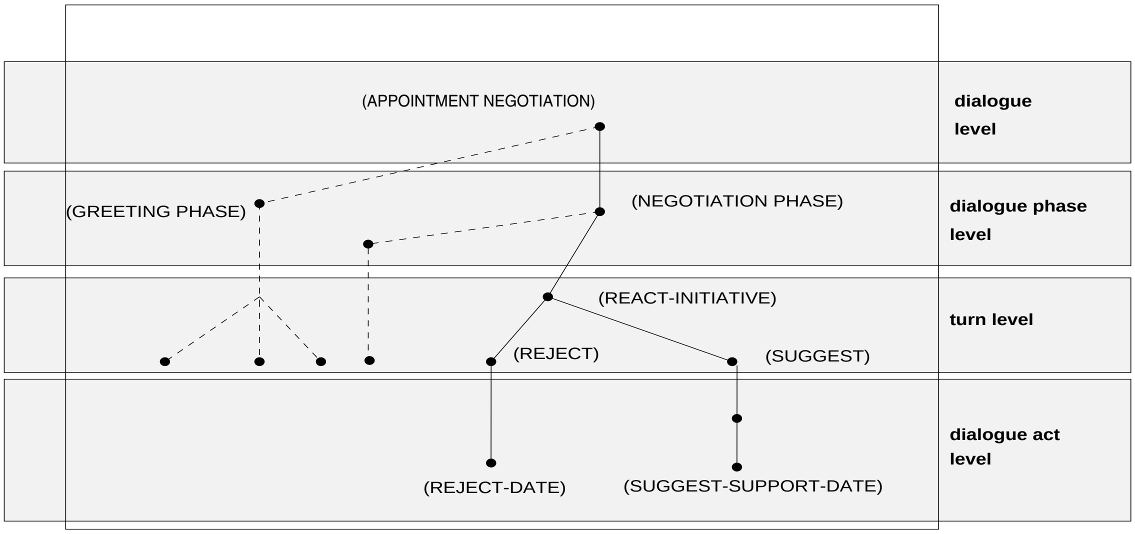

Figure 2: The four levels of the intentional structure

Plan Operators

Our plan operators follow the usual paradigm and decompose a goal into one or more subgoals. Pre-actions and actions can be defined which are evaluated when the operator is inserted in the tree and when all its subgoals are completed. It is also possible to define constraints to an operator which prevent it from being applied in certain contexts. An example of a plan-operator dividing the goal `DOMAIN-DEPENDENT SUGGEST ?INCONTEXT ?OUTCONTEXT]` into an operator processing iterations is shown below. The operator sets the last important utterance turn to suggest when completed, and modifyes the `?OUTCONTEXT` at the same time.

```
begin-plan-operator SUGGEST-OPERATOR
 pre-actions (nop)
 undo-pre-actions (nop)
 goal [DOMAIN-DEPENDENT
       SUGGEST
       ?INCONTEXT
       ?OUTCONTEXT]
 subgoals (sequence
           [IN-DOMAIN-DEPENDENT
            SUGGEST-ITER
            ?INCONTEXT
            ?OUTCONTEXT])
 actions (set-last-important-dialogue-act
          'suggest
          ?INCONTEXT
```

```
          ?OUTCONTEXT)
 undo-actions
        (unset-last-important-dialogue-act)
end-plan-operator
```

There are some special keywords like "sequence" used in the operator above. The operational semantic for this is: the subgoal(s) has (have) to be solved sequentially. It is also possible to use the key words "xor", "and", "iterate" and "optional" with their obvious operational semantics. The symbols beginning with a question mark are variables.

The plan recognizer operates on two single structures, the dialogue memory [6, 7], and the sequence structure. We do not, for efficiency reasons, keep multiple contexts so in the process of defining the plan operators we have to define undo-operations for both the "actions" and the "pre-actions" to reset the side effects performed by the actions/pre-actions when the intentional structure is backtracked away.

To increase the run time efficiency, we use two standard optimizations. The algorithm makes use of a tabulation mechanism, which stores a copy of the *complete* substructures that would be destroyed in case of backtracking. The copy can then be inserted when the recognizer tries to solve the same goal again. We also compile out so-called *reachable constraints* which prevents an operator to be applied when there is no possibility for its successful application[1]. This has turned out to be a very efficient, not to say necessary, tool for improving run time performance.

Repair

Exploring our first approach [2] we had to extend the plan recognizer to process unexpected input – *repair*. This is done by means of a set of specialized repair-operators together with some additional methods for inserting them into the tree. Dialogue acts like GIVE_REASON and CLARIFY_STATE have the property of appearing everywhere in the dialogues. Instead of writing an extra operator handling this, we use this repair mechanism for "repairing in" these dialogue acts.

A special problem are meta dialogues in the dialogue like *clarification dialogues*. They occur in (at least) two forms – pure clarification dialogues where one speaker makes a single CLARIFY_QUERY and the other speaker optionally responds with a single CLARIFY_ANSWER back. Unfortunately this is rare, and instead the clarification dialogues are embedded in the turns, which makes it very hard to mirror in the structure. Currently we process clarification dialogues by means of repair.

Problems with our first approach

It is a well known fact that man-machine dialogues and dialogues between two people that are allowed to speak freely, even in task oriented dialogues, differ in structure and, more importantly and not in favour to us, regularity. Two persons trying to convey the same goal, manage in our corpus to speak in very different ways, which makes the task of hand coding this behaviour very hard or even impossible.

[1] This corresponds to the improvement of a parser called "Top-down parser with Bottom-up filtering" as described in [14]

Our first version of the plan recognizer was based on hand coded plan-operators, and as an effect of the very different structures of the dialogues we processed, a lot of repair was needed. This in turn made the structure sometimes so awkward, that it was hard to observe any meaning from the structures.

3 Real World Data

Robustness

We here indicate some problems with the input to the dialogue module. Besides the well-known problem with speech recognition[2], we can observe the following problems:

Wrong segmentation A big problem with analysing spoken input is *segmentation*. Since a turn consists of not just one utterance, the turn has to be split into pieces. This process would be fairly simple if it was not for the fact that people do not speak grammatically correct, and, as pointed out above, problems with the speech recognition.

Missing deep information When the deep processing components fail to analyse, transfer or generate the turn, shallow methods are being used. These methods do not provide any deep information, but the dialogue act.

Multiple Dialogue acts The components analysing utterances in the VERBMOBIL system are currently just able to attach one dialogue act per utterance, although the utterance has more than just one function. This could make a machinery based on a hand-coded dialogue model fail.

To demonstrate what problems we are facing, we have taken an example from our corpus to show what consequences a missing multiple dialogue act can have[3]:

Example 3.1

PS1001a: `ja prima,` (FEEDBACK)
(Well)

PS1001b: `dann lassen Sie uns doch noch einen <!einn> Termin ausmachen.` (INIT_DATE)
(Let us make a date)

PS1001c: `wann wär's Ihnen denn recht?` (REQUEST_SUGGEST_DATE)
(When does it suit you?)

BS1002a: `also ich dachte noch in der nächsten Woche , auf jeden Fall noch im April <Klicken>.` (SUGGEST_SUPPORT_DATE)
(Well I thought in the next week, in any case in April)

[2]With a lexicon size of 3000 fullforms, the speech recognition components in the running system have a word accuracy of about 75%. This means that the actually spoken sentence is more or less guaranteed not to be found in the word lattice.

[3]The translation of the examples is not produced by VERBMOBIL.

PS1003a: `#Klicken# ja am Dienstag den sechsten April hätt' ich noch einen Termin frei allerdings nur nachmittags.` (ACCEPT_DATE)
(Yes on Tuesday the sixth of April is possible for me, but only in the afternoon)

PS1003b: `geht es da bei Ihnen <!Ihnn> auch <Klicken>?` (REQUEST_COMMENT_DATE)
(does that suit you?)

... □

Here the utterance **PS1003a**[4] is annotated with just ACCEPT_DATE, but it also has the function of proposing a new date (SUGGEST_SUPPORT_DATE). The following utterance **PS1003b** annotated with REQUEST_COMMENT_DATE can not, according to our dialogue model, follow an utterance of type ACCEPT – one should introduce a new topic before one can request a comment on, in this case, a date. There are different ways of coping with this, and as we will see, the approach proposed in this paper can contribute to handle these cases (see also [7][8]).

4 Using an Annotated Corpus

Because of the problems with hand coded plan operators, we were looking for alternative approaches, and since we have quite a big corpus of negotiation dialogues we were interested if it was possible to make use of it by automatically deriving plan operators from it. In this case, could they be used in the running system?

The VERBMOBIL corpus consists of 12 CD-Roms of transcribed German-German as well as German-English negotiation dialogues of which 5 are hand-annotated with dialogue acts. Currently, no time information has been annotated in the corpus. For the experiments described in this paper, we have used 3 CD-Roms of German-German dialogues, containing 276 dialogues for training and 50 German-English for testing.

Preparing the Corpus

The idea when deriving the plan operators is to classify the turns and then derive a set of operators for each class. We use a small amount of knowledge about the dialogue acts and the structure of the dialogues, which resulted in 9 classes (see below). Utilizing the knowledge about the dialogue acts, we first prepare the corpus. This is done in three steps:

1. First filter out parts of, or even utterances which do not contribute to the actual negotiation.

   Utterances which lie outside the scenario are attached with dialogue acts like GARBAGE, REFEER_TO_SETTING or DIGRESS_SCENARIO. Also some dialogue act types, which appear anywhere in the dialogues, like GIVE_REASON or CLARIFY_STATE, were filtered out from the utterances. They will be processed by means of the repair mechanism. The exceptions are turns consisting of just a CLARIFY_QUERY or a CLARIFY_ANSWER. They are detected and processed by hand-coded operators.

[4]Speaker **A** is indicated by **PS**n, and speaker **B** by **BS**n

2. The next step is to make use of our dialogue act hierarchy, and map the more special dialogue acts to the more general ones (cf. SUGGEST_SUPPORT_DATE is mapped to SUGGEST)

3. Finally we "collapse" iterative appearances of the same dialogue act to just one. This means that for instance a turn consisting of two utterances annotated with SUGGEST_SUPPORT_DATE and SUGGEST_EXCLUDE_DATE respectively is in the first abstraction step mapped to SUGGEST SUGGEST, which is is reduced to just SUGGEST. In this process we notice which dialogue acts appear more than once in a row, and correspondingly, we have to modify the set of dialogue acts implementing the dialogue act hierarchy.

Example 4.1

– `Original version`

MM4002a: `<P> <A> ich finde es zwar auch ganz schön` (FEEDBACK_ACKNOWLEDGEMENT)
(I find it also very nice)

MM4002b: `+/was da/+ <P> <hrm> daß wir da beim <P> <A> Sport unsere <!unsre> Besprechungen <!Besprechungn> gehalten haben <!ham>` (DIGRESS_SCENARIO)
(what there... that we talked during the last exercise)

MM4002c: `aber <A> ich finde doch daß wir auf die normalen <A> Werktage zurückgreifen sollten` (SUGGEST_SUPPORT_DATE)
(But I think we should aim for the regular working days)

MM4002d: `und uns das Wochenende für die Familie vorbehalten sollen,` (SUGGEST_EXCLUDE_DATE)
(and reserve the weekend for the family)

MM4002e: `<A> wie sieht es da <P> bei Ihnen bezüglich einer Besprechung montags denn <!enn> aus <#Störgeräusch> <P>` (SUGGEST_SUPPORT_DATE)
(What do you think about a meeting on Monday?)

`Filtered version`

MM4002: `aber <A> ich finde doch daß wir auf die normalen <A> Werktage zurückgreifen sollten und uns das Wochenende für die Familie vorbehalten sollen , wie sieht es da <P> bei Ihnen bezüglich einer Besprechung montags denn <!enn> aus <#Störgeräusch> . <P>` (SUGGEST)
(But I think we should aim for the regular working days and reserve the weekend for the family. What do you think about a meeting on Monday?)

□

In example 4.1 we see how a turn consisting of the sequence FEEDBACK_ACKNOWLEDGEMENT DIGRESS_SCENARIO SUGGEST_SUPPORT_DATE SUGGEST_EXCLUDE_DATE SUGGEST_SUPPORT_DATE is reduced to SUGGEST.

Applying the three steps to the corpus, containing a total of 3344 turns (of which 75, containing just digressions, were removed during step 1) yielded 357 different kinds of turns.

Class	Dialogue Acts
greet	introduce motivate_appointment greet
greet-initiative	reject accept motivate_appointment introduce init request_suggest request_comment greet suggest
initiative	suggest request_comment init request_suggest
initiative-response	reject suggest accept
response	motivate_appointment thank confirm reject accept
response-initiative	introduce motivate_appointment thank confirm init request_suggest request_comment accept reject suggest
unknown	init confirm accept reject motivate_appointment request_suggest request_comment suggest
bye	thank confirm accept bye
misc	request_suggest confirm motivate_appointment init request_comment accept thank bye greet reject suggest

Figure 3: The 9 classes and their corresponding dialogue acts

Deriving Plan Operators from the corpus

The next step in the process of deriving plan operators from the corpus is to classify the turns. Examining a part of the corpus, we found that most of the turns fit into one of the following 9 classes[5]: *greet, greet-initiative, bye, response, response-initiative, initiative, initiative-response, unknown,* and *misc*. In figure 3 we see the classes and their corresponding dialogue acts. The class *initiative-response* is worth a comment – we found 12 turns where the speaker suggested something and rejected or accepted the suggestion herself.

When dividing the corpus we respected all turn patterns occurring more than once in the corpus.

The Learning Algorithm

For deriving the plan operators we have used the Bogus system [10, 11]. It is a system for deriving, for instance, *Hidden Markov Models* and *Stochastic Context Free Grammars*. It is based on an approach to the learning problem of probabilistic language models, known as "Bayesian Model Merging", and has some properties we want to utilize. Since we are interested in real time performance we can, during runtime, take advantage of statistic information generated by the system. An example of the result derived by the system is given in the appendix.

5 Evaluation

Overall Evaluation

We tested the derived plan operators on 50 dialogues with the following results: From 531 turns in total, we managed to classify 471 (88.7%) turns. Of these 471 two thirds (67.2%) were in the training corpus and correctly classified. Of the remaining

[5] It has, as pointed out in the outlook, to be further investigated if these classes are sufficient.

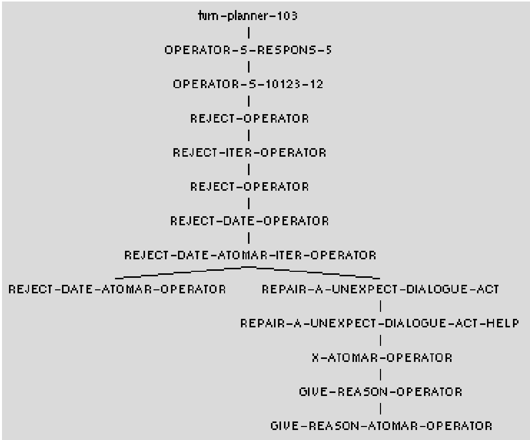

Figure 4: After `reject_date` and `give_reason`

155 (32.8%), 93 were correctly classified, and 62 were incorrectly classified. This means that a total of 409 of the 471 (86.8%) was correctly classifyed. Of the remaining 62, about 50% were involved with repair.

Detailed Example

We now take look at a running example taken from one of the test dialogues. We show how the plan recognizer incremetally builds a structure given the following turn:

Example 5.1

THW002a: `das <P> ist zu knapp` (REJECT_DATE)
(This is not enough time)

THW002b: `weil ich <P> <:<#> ab:> dem dritten in Kaiserslautern bin` (GIVE_REASON)
(because I will be in Kaiserslautern from the third)

THW002c: `<#> genaugenommen nur am dritten` (SUGGEST_EXCLUDE_DATE)
(to be precise just on the third)

THW002d: `<A> wie wäre es denn <P> am<Z> <P> +/F=/+ <ähm> <:<#> Samstag , dem:> zehnten Februar?` (SUGGEST_SUPPORT_DATE)
(How about on... – ehm – Saturday the tenth of February)

□

In figure 4 we see a screen-dump of the intentional turn structure after processing the first two utterances. The recognizer assumes that the turn is a response-turn (OPERATOR-S-RESPONS-5). The GIVE_REASON is repaired into the existing tree. The operators with -ATOMAR- in their names are unified with the information in the dialogue memory, whereas the operators with -ITER- are processing iterative occurrences of the same subgoal.

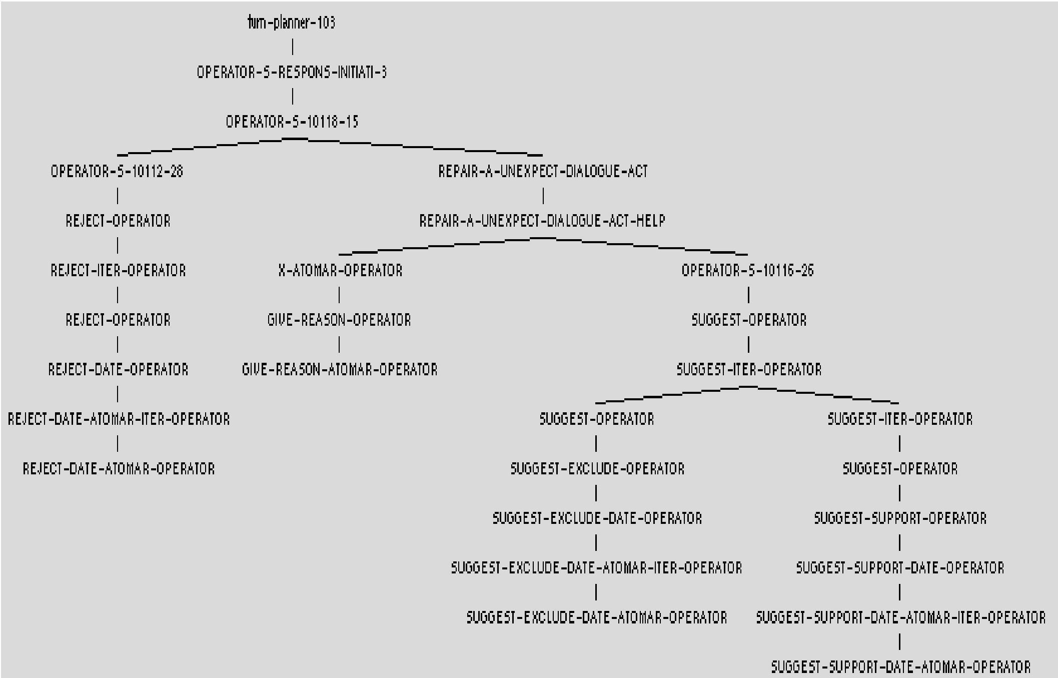

Figure 5: After SUGGEST_EXCLUDE_DATE

Figure 5 shows the structure after the third utterance, `suggest_exclude_date`, is processed. The top goal of the turn has changed into a response-initiative. Observe that the GIVE_REASON now is attached to the SUGGEST_EXCLUDE_DATE – it is impossible without information about the propositional content of the utterances to determine where it should be attached, so currently this type of repair is done in a kind of ad hoc manner.

Finally, the second suggestion SUGGEST_SUPPORT_DATE is added to the tree to the topmost domain-dependent suggest-operator (see figure 6). This operator was modified to handle iterative appearances of utterances with the same dialogue act.

6 Outlook

We have presented a method for the semiautomatically derivation of plan operators from a hand-annotated corpus of dialogue act sequences. The results of the first evaluation has yielded quite promising results. We will continue exploring these ideas. There are, however, some tasks which need more work:

- The derived plan operators do not cover 100% of the test corpus. Should this be taken care of by more sophisticated repair methods?

- Are the classes optimally chosen? – should we look at subclasses and build some classes on top of them. It is tempting to divide the classes concerned with *response* and *initiative* into subclasses and then use them to construct the classes response-initiative and initiative-response based on them.

  Moreover, the classes chosen are a little bit to coarse-grained. They do not, for instance, respect giving the initiative away (REQUEST_SUGGEST_DATE), which for instance could make it possible to resolve some of the turns in the category "unknown".

- How big must the training set be before we get a sufficient coverage?

- To even more speed up the analysis process, we can use the prediction mechanism provided by the statistical component, and predict what class(es) is (are) most probable to come next.

  This could also be used to predict language models for the speech recognition components to improve the recognition rates.

- Is the learning system suitable for our purposes? One advantage and at the same time risk with this approach is that the derived grammar might overgenerate, i.e. it is possible to recognize input which is not in the training set. A good guess is that a grammar like the one yielded by the BOGUS system is a more compact and efficient representation than just the naive grammar constructed by letting the top symbol of the grammar expand into exactly every sentence in the training set.

References

[1] Jan Alexandersson, Elisabeth Maier, and Norbert Reithinger. A Robust and Efficient Three-Layered Dialog Component for a Speech-to-Speech Translation System. In *Proceedings of the 7th Conference of the European Chapter of the ACL (EACL-95)*, pages 188–193, Dublin, Ireland, 1995. also available as Verbmobil Report Nr. 50, DFKI GmbH, Dezember 1994. available in the cmp-lg electronic archive under no. cmp-lg-9502008.

[2] Jan Alexandersson and Norbert Reithinger. Designing the Dialogue Component in a Speech Translation System – a Corpus Based Approach. In *Proceedings of the 9th Twente Workshop on Language Technology (Corpus Based Approaches to Dialogue Modelling)*, Twente, Holland, 1995.

[3] John Austin. *How to do things with words.* Oxford: Clarendon Press, 1962.

[4] Susanne Jekat, Alexandra Klein, Elisabeth Maier, Ilona Maleck, Marion Mast, and J. Joachim Quantz. Dialogue Acts in VERBMOBIL. Verbmobil Report 65, Universität Hamburg, DFKI Saarbrücken, Universität Erlangen, TU Berlin, 1995.

[5] Martin Kay, Jean Mark Gawron, and Peter Norvig. *Verbmobil. A Translation System for Face-to-Face Dialog.* Chicago University Press, 1994. CSLI Lecture Notes, Vol. 33.

[6] Elisabeth Maier. A multi-dimensional representation of context in a speech translation system — a practical approach. In *Proceedings of the IJCAI-95 Workshop Context in Natural Language Processing*, Montreal, August 1995. also available as VERBMOBIL Report No. 64.

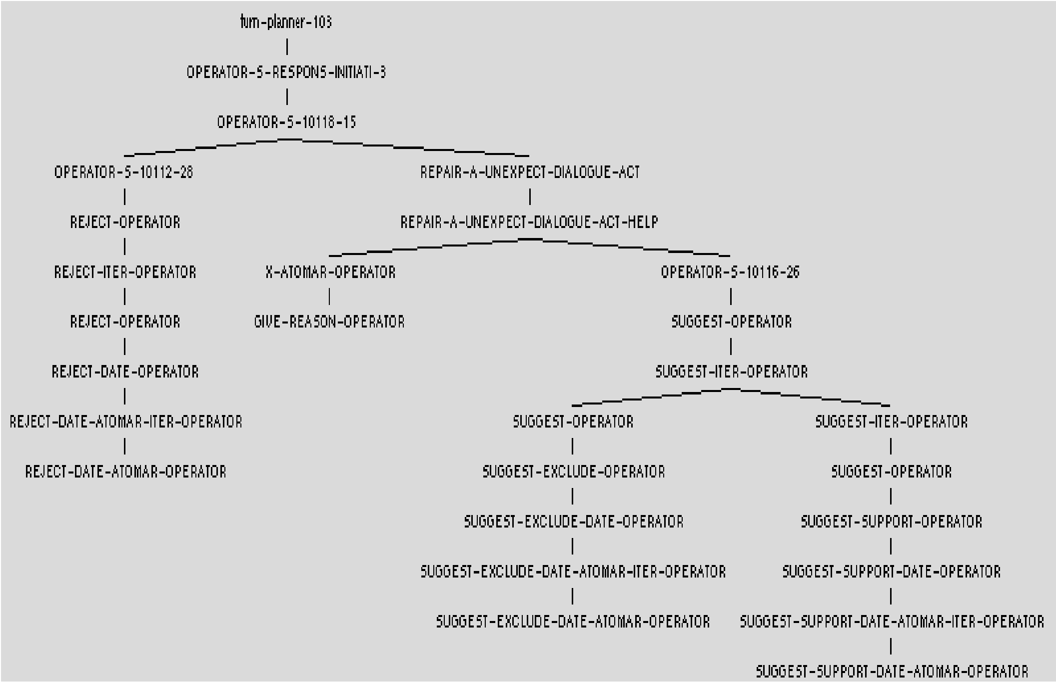

Figure 6: After SUGGEST_SUPPORT_DATE

[7] Elisabeth Maier. Context Construction as Subtask of Dialogue Processing - the VERBMOBIL Case. In Anton Nijholt, Harry Bunt, Susann LuperFoy, Gert Veldhuijzen van Zanten, and Jan Schaake, editors, *Proceedings of the Eleventh Twente Workshop on Language Technology, TWLT, Dialogue Management in Natural Language Systems*, June 19-21 1996.

[8] Elisabeth Maier and Norbert Reithinger. Utilizing statistical speech act processing in verbmobil. Technical report, DFKI, December 1994.

[9] J.R. Searle. Indirect Speech Acts. In P. Cole and J.P. Morgan, editors, *Syntax and Semantics*, volume 3: Speech Acts. Academic Press, New York, 1975.

[10] Andreas Stolcke. *Bayesian Learning of Probabalistic Language Models*. PhD thesis, University of California at Berkley, 1994.

[11] Andreas Stolcke. *How to Boogie: A Manual for Bayesian Object-oriented Grammar Induction and Estimation*. International Computer Sience Institute of Berkley, California, 1994.

[12] Marc B. Vilain. Getting Serious about Parsing Plans: a Grammatical Analysis of Plan Recognition. In *Proceedings of AAAI-90*, pages 190–197, 1990.

[13] Wolfgang Wahlster. Verbmobil–Translation of Face-to-Face Dialogs. Technical report, German Research Centre for Artificial Intelligence (DFKI), 1993. In Proceedings of MT Summit IV, Kobe, Japan, July 1993.

[14] Mats Wirén. *Studies in Incremental Natural-Language Analysis*. PhD thesis, Department of Computer and Information Science Linköping Institute of Technology, 1992.

Appendix

We here give an example of a class and its corresponding grammar. The number means that the pattern occurred that many times, and the parenthesized symbols means that the symbol occurred more than once in a row.

```
;; data greet-initiative

(5 (introduce_name) (suggest))
(5 greet (introduce_name) (suggest))
(4 (greet) (suggest))
(3 greet (introduce_name) (suggest) request_comment)
(3 greet (introduce_name) (suggest) request_suggest)
(3 (greet) (init) suggest request_comment)
(3 greet (introduce_name) init (suggest)
   request_comment)
(3 greet (introduce_name) init (suggest))
(3 greet (introduce_name) (init))
(3 (greet) (init) (suggest))
(2 greet (introduce_name) init suggest
   request_suggest)
(2 introduce_react introduce_name (suggest))
(2 (greet) init request_suggest)
(2 greet (introduce_name) motivate_appointment
   init suggest)
(2 greet (request_suggest))
(2 greet motivate_appointment (init))
(2 greet introduce_name (accept) suggest)
(2 (greet) init clarify_query)
(2 greet (introduce_name) init
   motivate_appointment)
(2 greet (introduce_name) motivate_appointment
   init request_suggest)
(2 introduce_name motivate_appointment (suggest)
   request_comment)
(2 greet introduce_name motivate_appointment
   (request_suggest))
(2 greet introduce_name motivate_appointment
   (suggest))
(2 (introduce_name) clarify_query)
(2 (introduce_name) init suggest)
(2 greet (introduce_name) request_suggest)
(2 greet introduce_name reject (suggest))
(2 greet (introduce_name) init request_suggest)
(2 greet (introduce_name) clarify_query)
(2 (greet) init)
```

For the grammar, the leftmost position of the right hand side of a rule is a tuple which should be read *probability;no-of-occurrences* – this information can be used during runtime to choose the operator with the highest probability.

```
;; grammar greet-initiative

((0 ->  (1.536e-2;2.00000    1 36)
        (1.536e-2;2.00000    1 59 2 36)
        (1.536e-2;2.00000    1 69 2)
        (1.280e-3;1.00000    1 69 59 2)
        (4.353e-2;4.00000    58 36)
        (7.170e-2;6.00000    64)
        (4.353e-2;4.00000    65 2)
```

```
       (1.280e-3;1.00000    65 87)
       (0.410;30.0000       87)
       (0.367;27.0000       87 36)
       (1.536e-2;2.00000    87 59))
(58 -> (1.000;31.0000      65 1))
(64 -> (0.875;18.0000      1 2)
       (0.125;3.00000      59 69))
(87 -> (0.134;12.0000      58 59)
       (0.170    ;15.0000     58 69)
       (3.787e-2;4.00000     65 36)
       (0.170    ;15.0000     65 64)
       (0.158    ;14.0000     65 69)
       (0.279    ;24.0000     87 2)
       (4.991e-2;5.00000     87 69))

(1 ->  (1.000    ;56.0000     . introduce_name))
(2 ->  (1.000    ;51.0000     . suggest))
(36 -> (0.144    ;6.00000     . clarify_query)
       (0.306    ;12.0000     . request_comment)
       (0.550    ;21.0000     . request_suggest))
(59 -> (7.407e-2;2.00000     . accept)
       (0.852    ;16.0000     . motivate_appointment)
       (7.407e-2;2.00000     . reject))
(65 -> (0.963    ;66.0000     . greet)
       (3.676e-2;3.00000     . introduce_react))
(69 -> (1.282e-2;1.00000     . feedback)
       (0.987    ;39.0000     . init)))
```