

A Pragmatics-First Approach to the Analysis and Generation of Discourse Relations

Bernd Ludwig, Heinrich Niemann, Martin Klarner, Peter Reiß, Günther Görz

University Erlangen-Nürnberg

Bavarian Research Cooperation

for Situated, Individualized, and Personalized Human-Computer Interaction

Haberstraße 2, D-91058 Erlangen, Germany

Abstract

In this paper, we present some key issues of our dialogue system for spoken language¹ which has been designed to enable quick configurability to various applications and to apply deep syntactic and semantic analysis, discourse processing, and language generation. It features a complex semantics-pragmatics interface in the sense of (Brietzmann and Görz, 1982): Discourse and application pragmatics are considered independent from each other; user utterances affect application pragmatics in that they are translated into speech acts which, in turn, indicate a user request to the application. The system's behaviour is specified in a proprietary programming language interpreted by the dialogue manager. The semantics of this language is defined in terms of an extended version of DRT.

1 Introduction

Rational dialogues which are guided by GRICE's maxims of conversation serve as a communicative tool in jointly executing tasks in the application domain. We compute plans (chronologically ordered sequences of atomic actions) that devise a way to complete these tasks.

¹Acknowledgement: Our work is supported by the Bavarian Research Association FORSIP

The interpretation of new contributions and their integration into the dialogue is controlled by global factors (e.g. the assumption that all dialogue participants behave in a cooperative manner) and by local factors (e.g. the current state of plan execution). Within this framework, the paper focuses on two issues: First, how can one determine whether an utterance contributes to executing the current plan? Second, how do the results of such an analysis affect the reaction of the dialogue participant?

Our example scenario is an online B2B shop (Fischer et al., 2002) where one can buy boxes of different types, sizes, colours, and materials. This shop offers various functions (*transactional tasks*) like searching the data base, selecting articles and moving them to the shopping cart. An offer can be requested, the price be negotiated, and finally the user accepts or rejects the offer. A typical dialogue is shown in figure 1.

2 The Dialogue System

The processing of utterances in our dialogue system begins with passing them to a speech recognizer. Then a syntax parser (Bücher et al., 2002) computes the semantic representation for the utterance. The dialogue manager handles the negotiation between the user and the online shop.

2.1 Natural Language Understanding

The parser works in two phases: First, it segments the output of the speech recognizer into chunks (Abney, 1991). Second, these chunks are translated into constraints on a (partial) description of

System₁: Guten Tag und willkommen in unserem Shop! Was kann ich für Sie tun? (Hello and welcome to our shop! How may I help you?)
User₁: Ich suche Boxen in Rot. (I am looking for red boxes.)
System₂ (highlights an item in a list of proposed products): Ich empfehle Ihnen diesen Artikel, denn er kommt Ihren Präferenzen am nächsten. (I recommend this product, as it provides the best match for your preferences.)
User₂: Okay.
System₃: Hier sehen Sie ihren Warenkorb. Soll ich Ihnen dafür ein Angebot erstellen? (Here is your shopping cart. Shall I make you an offer for it?)
User₃: Ich möchte meine alten Aufträge sehen. (I want to see my previous orders.)
System₄: Bitte schön. (Here you are.)
User₄: Bitte mach mir ein Angebot! (Make me an offer, please!)
System₅: Ich kann Ihnen die Waren für 734 Euro 90 anbieten. Der Rabatt beträgt 6%. (May I offer you the shopping cart contents for 734 Euro 90? The price is 6% off the retail price.)
User₅: Das ist mir zu teuer. (This is too expensive.)
System₆: Ich kann Ihnen noch um 20 Euro entgegenkommen. (I can give you 20 Euro additional discount.)
User₇: Ich nehme das Angebot an. (I accept your offer.)
System₈: Sie haben einen guten Kauf getätigt. Vielen Dank. (You have made a good deal. Thank you.)

Figure 1: An example dialogue.

the desired system state. For that purpose, an approach motivated by dependency theory is applied, in which valencies of the syntactic head of each chunk are analyzed if they can serve as the dependent of some other chunk (its regent). Dependent and regent have to meet three classes of criteria in parallel: syntactic, semantic (is the semantic part of the valency satisfied?), and pragmatic constraints (can a constraint in the application domain be derived from the triple "regent", "thematic role", and "dependent"?). For the utterance "I am looking for red boxes." (see figure 1) the parser computes the following informal information:

<i>Chunk</i>	<i>Semantics (informal)</i>
<i>I</i>	customer
<i>am looking for</i>	customer's task
<i>red boxes</i>	involved object(s)

Based on that analysis, the DRS in figure 2 is derived as a partial description of the user's intention.

Parallel to the analysis of the content, a speech act is assigned to the utterance in order to determine its discourse-pragmatic function. The example utterance indicates a *request* to the application.

<i>o s t p q f a</i>	Comment
Offer(<i>o</i>)	expected effect of customer's task
has-status(<i>o, s</i>)	
Confirmation(<i>s</i>)	object involved in the task
has-catalog(<i>o, t</i>) T-New-Catalog(<i>t</i>)	
has-product(<i>t, p</i>)	
Product(<i>p</i>)	
has-article(<i>p, a</i>)	
Article(<i>a</i>)	
has-feature(<i>a, f</i>) feature(<i>f</i>)	
has-FNAME(<i>f, CAA074001</i>)	
fname(CAA074001)	
has-FVALUE(<i>f, red</i>) fvalue(red)	

Figure 2: Pragmatics of a user utterance.

In order to plan an applicable system reaction to the utterance, the dialogue system first needs to find out whether there is a plan that satisfies the request. In our terminology, the request constitutes an *interactive task* that triggers a *transactional task*, as discussed in the following section.

2.2 Plan Generation and Execution

The content of each transactional task is viewed as a planning goal. If a plan for it can be constructed, the system reacts by executing the plan and monitoring the success of each step performed. To compute plans, we have configured the FF Planner (Hoffmann and Nebel, 2001) with plan operators that match the application. The transactional tasks that the shop performs are formalized in terms of PDDL plan operators. This approach can be generalized: All applications whose functionality can be expressed in PDDL are suited for integration into the outlined dialogue model.

After computing a plan, each atomic action in it is executed (figure 3 shows the plan for the goal in figure 2). Step by step, the dialogue manager sends atomic actions to the shop agent and requests them to be executed. In order to verify

```

1: PRODUCTSEARCH N U1 TB B SVEN
2: REQUEST S SVEN TS1
3: ADDTOSHOPPINGCART S SVEN TB TS1 BA
4: QUERY-IF R YES SVEN
5: OFFERREQUEST N TS1 O1 SVEN
6: OFFERACCEPTANCE N TS1 O1

```

Figure 3: A plan representing a complex user command. (Transactional tasks are written in typewriter font, interactional tasks in **boldface**.)

whether the plan can actually be completed, the action's effects on the application are compared with those expected in the plan. If they diverge, a recovery strategy is initiated.

2.3 When Is a User Request Satisfied?

To keep track of the current state of a dialogue, we implemented an extended version of DRT that provides an explicit formal representation of the currently pending requests. Delegation of actions to be executed requires involved agents to give feedback about the outcome. Based on this feedback, the dialogue manager updates the processing state of each user request (i.e. it concludes if a request is satisfied or still being processed). The central idea is that a user request is satisfied if the plan assigned to it by the dialogue manager has been processed completely. However, in many cases the execution of some action may fail. In such a case, the shop agent provides the dialogue manager with an explanation about the failure in terms of a DRS that contains the preconditions of the action in the current application situation that are inconsistent with the preconditions computed in the plan. With this information, the dialogue manager tries to continue the dialogue in a way that provides as much information as possible.

As it will be discussed in more detail in section 3.2, the dialogue manager's plans themselves about how to interact with the user can fail. Incoherent input from the user is an important cause for that, but the failure of a plan for a transactional task may force the dialogue manager as well to re-plan the interaction with the user.

3 Discourse and Application Pragmatics

The plan in figure 3 highlights the fact that the dialogue model presented here draws a clear distinction between transactional and interactive tasks. While the first type of tasks aims at an explicit representation of the shop's transactions, the second represents tasks the dialogue system needs in order to start the interaction with the user when the execution of a plan to control the shop requires to do so.

3.1 Processing of Interactive Tasks

The interactive tasks are carried out by the dialogue system itself; they can be specified in detail in a proprietary programming language that allows for mapping of tasks onto speech acts which, in turn, depends on the current dialogue situation (see section 3.2). Interactive tasks determine the way in which the user contributes to the construction or execution of a plan. In general, there are three different types of contributions and thus three different classes of interactive tasks: *Modifying* information, *querying* information, and *execution* of actions. In cooperative dialogues, interactive tasks are completed when certain expectations depending on the class a task belongs to are met: For *modification*, one has to verify whether it can be performed without conflicts with the available knowledge. For *querying*, the queried information has to be computed. For *execution*, it has to be verified whether a plan can be found and executed. Depending on the type of information in the content of the interactive task – e. g. the intensional or extensional knowledge a task refers to – the knowledge about plan operators, terminological definitions, linguistic information, or the knowledge about the current dialogue and application situation can be the topic of utterances in a dialogue. In the class of dialogues discussed in this paper, interaction with the user is limited to the current situation, i. e. the system cannot learn new rules about the application.

In such an approach to dialogue, natural language utterances fulfill two functions: On the one hand, they indicate an interactive task (e. g. by cue phrases, questions, or imperatives). On the other hand, they indicate a transactional task or information about the current situation.

The function of an utterance in the current situation must be specified in order to integrate it in a dialogue hypothesis. For this purpose, the preconditions of the hypothesis are verified both in the dialogue situation for the interactive task and in the application situation for the transactional task. Thus, the main claim of this paper is that for the interpretation of rational dialogues it is necessary to distinguish between the current state of interaction (*the dialogue situation*) and the current state of the application task (*the application situation*).

3.2 Acting and Reacting in a Dialogue

A major implication of the described approach is that plans have to be computed for interactive tasks as well as for transactional ones. A plan for an interactive task determines the *reaction* to the new contribution (see figure 4): For building such a plan, operators that encode the global factors mentioned in the introduction are applied.

The next step in reacting to the user utterance is to execute the plan. Finally, if the execution succeeded, the completion of the reaction is signalled by ACCEPT. Like REQUEST and QUERY-IF in figure 3, ACCEPT is a basic dialogue operation that maps interactive tasks onto (sequences of) speech acts and determines the content of utterances to be realized by the text generation module (see section 7).

Which speech acts are generated to verbalize the interactive task depends on the requirements of the dialogue situation. In human communication, the decomposition of (discourse) pragmatic goals into speech acts and their verbalization is influenced to a large extent by a number of factors like topicalization, stylistic variability, relations between dialogue participants, availability and limitations of resources, cognitive capacity, and personality of the dialogue participants (see (Carolis et al., 2000; Cassell et al., 2000)). In order to consider these factors when generating speech acts, all dialogue operations are programmable.

The implementation of ACCEPT in figure 5 considers limitations of the cognitive capacity of the hearer, by summing up all discourse referents t in `contribution` and computing the relevance of the contribution's content and its amount. By relating the amount to the temporal distance to the last utterance in the dialogue, a "information speed" is calculated. If it is too high, no speech act will be generated unless ACCEPT is important enough.

Our example shows that in order to reach a

```
1: FIND-PLAN BUY P
2: EXECUTE-PLAN P
3: ACCEPT BUY
```

Figure 4: Discourse plan for our example. (Interactive tasks are written in **boldface**.)

```
proc ACCEPT(task contrib);
drs utterance = content(contrib);
float rel=0.0, vol=0.0, speed, sda_dist;
begin
  forall dr t in utterance do begin
    vol := vol+consumption(utterance,t);
    rel := rel+priority(utterance,t)
  end;
  sda_dist := time() - last_sda_time;
  last_sda_time := curr_time;
  speed := vol/sda_dist;
  old_speed := speed
  if speed < 0.25 then utter contrib;
  else if rel > 4.0 then utter contrib;
end;
```

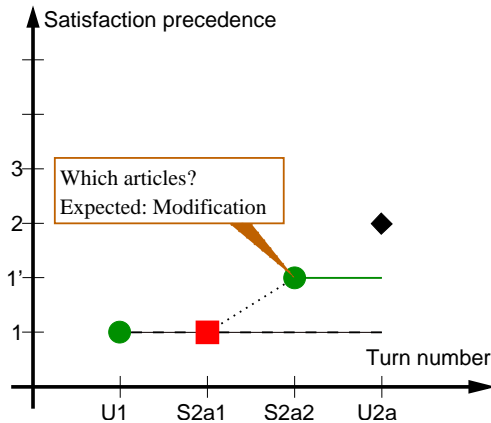
Figure 5: Implementation of ACCEPT

discourse-pragmatic goal, factors not determined by the content of a task (e. g. information speed) have to be taken into account. This issue becomes even more complex when the system is requested to explain the pragmatic reasons that lead to an utterance, e.g. to explain in detail why a transactional task or an atomic action failed. Such a behavior would be important in instructional dialogues, but it could be wrong or even dangerous when a plan has to be executed as fast as possible.

4 Building up the Discourse Structure

The preceding section explained that discourse structures result from the execution of discourse plans. The discourse structure is a graph whose edges relate interactive tasks (the nodes of the graph). As in (Grosz and Sidner, 1986), an edge between two nodes determines in which way the later interactive task contributes to the completion of the earlier one. From the viewpoint of the discourse model presented in this paper, the *satisfaction-precedence* relation holds between two tasks if they are subsequent steps in a plan. For instance, QUERY-IF (step 4 in figure 3) is *satisfaction-preceded* by REQUEST (step 2): After the online shop has found articles in its data base and presented them to the user (step 1), it requests the user to select some of them to put on the shopping cart (step 2). The shop cannot ask the user if he wants an offer (step 4) until steps 2 and 3 are completed.

The *dominance* relation holds between two tasks if the dominated task contributes to the com-



A square stands for a failure in a plan; circles for new user utterances, diamonds for system reactions to a blocking, and lines for discourse relations.

Figure 6: Chart of discourse relations after new turn has been shown irrelevant to the current focus

pletion of the dominating one. For example, all user responses are dominated by interactive tasks initiated by the system. Figure 6 shows the discourse structure after step 2 in figure 3. *Satisfaction-precedence* is visualized by going up the y axis, while *dominance* holds between tasks if they are crossed by the same parallel line along the x axis. The goal of an interactive task is achieved if it dominates another one. In this case, its expectation is met. If the expectations of an interactive task are not met, consider this modified continuation of the example dialogue (see figure 1):

User₁ (U1): *I am looking for red boxes.*

System_{2a1} (S2a1): You may now choose from the articles in the presented selection.

System_{2a2} (S2a2): Which one would you like?

User_{2a} (U2a): *I need containers.*

In the example above, even the reaction of the dialogue system (utterance **System_{2a}**) violates the expectation assigned to the interactive task (**User₁**) that has been initiated by the user (see figure 1). In order to react appropriately, the user has to analyze the system's utterance and understand that it indicates an admissible step in a plan for his request **User₁**. Analogously, the system has to find

out whether user utterances constitute expected reactions that allow the execution of the next step in the plan.

5 Logical and Discourse Relations

Assuming that boxes and containers are two different types of articles and that therefore the user does not choose any article from the proposed selection with **User_{2a}**, the goal for **User_{2a}** is incompatible with the plan for **User₁**. Neither *satisfaction-precedence* nor *dominance* holds between **System_{2a}** and **User_{2a}**.

In a situation like this it is useful to analyze relations that eventually hold between the transactional tasks assigned to **User₁** and **User_{2a}**. The implementation of the online shop may allow other relations than *satisfaction-precedence* which implies that the first task should be completed before the second one can start. If the shop is capable of presenting multiple windows for several selections at the same time, it can handle two or more transactions concurrently.

Concurrent execution of plans can be modelled with the help of the planner presented in section 2.2. Actually, the planner computes a partial order of actions that need to be executed to achieve the corresponding goal: A step in the plan may imply the execution of one or more actions; in the latter case, all the specified actions will be executed in parallel. Consequently, if one wants to test whether two interactive tasks are connected by the relation *in-parallel*, one has to compute a plan for the conjunction of the associated transactional tasks. If there is a plan, *in-parallel* eventually holds (see section 6).

Otherwise, **User_{2a}** *blocks* the execution of the current plan. In this case, neither *in-parallel* nor *satisfaction-precedence* or *dominance* correctly express the relationship between interactive tasks. As (Asher and Gillies, 2003) observes, additional discourse relations are needed if a conflict occurs when integrating an utterance into the discourse structure to adequately describe it.

In contrast to (Asher and Gillies, 2003), our approach distinguishes between discourse relations (which statically describe a state of affairs) and discourse pragmatic plans (which devise a way to proceed a dialogue): If it turns out during the anal-

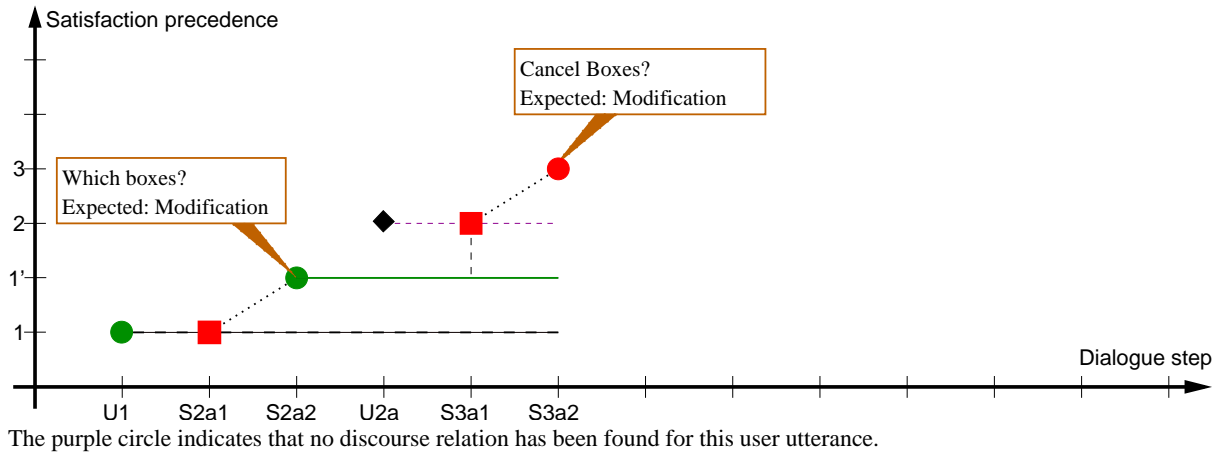


Figure 7: Discourse structure resulting from a *block* relation between the last utterances

ysis of **User**_{2a} that the new contribution blocks the completion of the current plan, this conflict must be solved. In figure 7, the discourse structure is shown for the following dialogue continuation:

System_{3a1} (S3a1): You cannot browse boxes and containers at the same time.

System_{3a2} (S3a2): Do you want to stop selecting boxes?

In our analysis **System**_{3a1} is *dominated* by **User**_{2a} as it completes the associated interactive task (by indicating failure). **System**_{3a2} *satisfaction-precedes* all utterances, since without an answer, the system cannot decide how to complete the pending transactions. In ASHER's terms a *correction* relation would hold between dialogue step **User**_{2a} and **System**_{3a1}. In our view, however, CORRECTION is the underlying interactive task (see figure 7). CORRECTION has been performed because an adequate discourse-pragmatic plan was found, which allowed the dialogue system to ask the user for information; this is done *in-parallel* to the pending query **System**_{2a2} in which the user is asked to choose from the presented items in the shopping cart. Basically, ASHER's and our analysis do not differ; however, the distinction between relations and tasks is better suited for an efficient implementation of a dialogue model that allows for analysis *and* generation of dialogue turns.

The user can react to the system's attempt to resolve the blocking of the plan by saying

User_{3a} (U3a): Yes.

This results in cancelling the blocked plan (the discourse structure related to this decision is shown in figure 8) which enables the dialogue system to complete its plan. Now, a plan that satisfies the transactional task of **User**_{3a} can be computed and executed (see **System**₃ in figure 1).

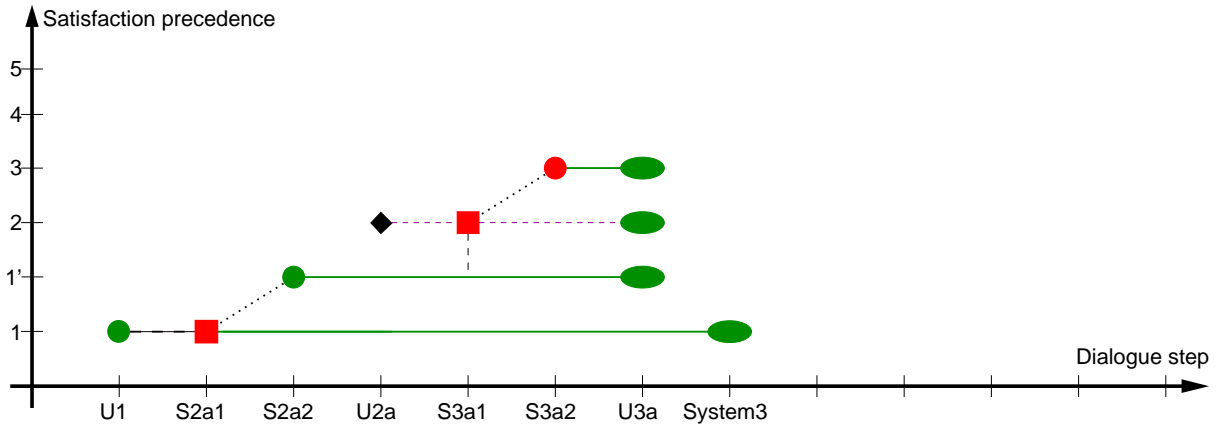
6 Deciding how to React

Depending on certain constraints in the application situation, when expectations of interactive tasks are not met, the logical relation between the associated transactional tasks may lead to an ambiguous discourse relation between a new contribution and the previous dialogue.

In this case, a decision about which of the possible interpretations to choose has to be made. This decision is based on a valuation of the options in the current situation. Figure 9 shows how these options are scored. Three factors are taken into account: First, the type of the interactive task $\text{expr}(H)$ does not meet the expectation $\text{evoc}(A)$ of the task A whose plan is currently executed. Second, the transactional tasks associated with A and H can be executed concurrently, and, third, A should be a good anchor for H (ideally, A is the current focus). Based on this computation, the following decision rule is applied:

$$\text{relation}(H, A) = v \leftrightarrow v = \text{argmax}_x V(x|H, A) \\ x \in \{\text{in-parallel}, \text{Blocking}\}$$

This rule selects the hypothesis with the best valuation. If there is no unique v , then a



An ellipsoid stands for an utterance signalling the interactive or transactional task at the start of the line to be satisfied.

Figure 8: Discourse structure after the blocking of the plan has been resolved

$$\begin{aligned}
 V(\text{in-parallel}|H, A) &= V(\text{expr}(H) \neq \text{evoc}(A)|H, A) \cdot \text{Does task type match expectations?} \\
 &\quad V(\text{parallel}(A, H)|H, A) \cdot \text{Can transactions run in parallel?} \\
 &\quad V(\text{anchor}(A, H)|H, A) \cdot \text{Is A a good anchor for H?} \\
 V(\text{Blocking}|H, A) &= V(\text{expr}(H) \neq \text{evoc}(A)|H, A) \cdot V(\text{blocked}(A, H)|H, A) \cdot V(\text{anchor}(A, H)|H, A)
 \end{aligned}$$

Figure 9: Valuation of discourse relations

CLARIFICATION is started which generates a query in which the user is asked to choose an alternative:

System_{3a3}: Do you want to browse boxes and containers at the same time?

This example shows that blocking in a discourse pragmatic plan is handled in analogy to the blocking of a transactional task. The dialogue model allows that information from the dialogue situation becomes the topic of utterances and discourse pragmatic tasks are associated to interactive ones.

7 Verbalizing the Discourse Structure

After building the DRS for **System_{3a3}**, it must be communicated to the user. This is done by a separate natural language generation (NLG) component which converts an input DRS (encoded in XML) into German sentences. In principle, this task is the reverse of analyzing user utterances; in practice, however, we have decided to implement the NLG component in a different way. Nevertheless, linguistic and pragmatic resources initially designed for analysis are used in our NLG system as well. Our approach to NLG relies on a hybrid combination of canned text, shallow, and deep

generation (Klärner and Ludwig, 2004). It uses templates which can also be generated by the deep generation branch, an extended version of bottom-up generation (van Noord, 1990). Once generated, the templates are stored for further use by the shallow generation module. The generated template for **System_{3a3}** is displayed in figure 10. It contains a `Token` with an `XPath` entry matching the appropriate part of the XML input structure and a bracketed reference to this `Token`. The parts in brackets indicate a possible enumeration of direct objects in the template sentence.

8 Related Work and Conclusions

The presented dialogue system differs from others in three key issues: First, planning and processing of discourse and application are kept separate. This allows for a plug-and-play architecture. Consequently, the functionality of the application can be modified without affecting the discourse model and vice versa. In (Allen et al., 2001) an architecture is described that integrates planning and discourse more closely and therefore seems to be less flexible. In systems as (Lamel et al., 1998), a dialogue strategy is hardwired in the di-

```

DialogAct: query_yesno
Token #1:      "../CHARACTERISTICS[CNAME="Usage"]/CVALUE/text()"#N.kasus_AKK/text()"
Template #1: "Do you want to browse (%1) at the same time?"

```

Figure 10: Generated template for a system utterance

dialogue manager's kernel. In our system, however, a dialogue designer may modify an existing strategy or even add new strategies during configuration. In this way, non-linguistic factors of interaction (e. g. user emotions) may influence the system's dialogue behaviour. Whereas these aspects are discussed in isolation in the literature, our system provides a platform for integrating them under real-time conditions. Second, our system applies deep processing to parsing as well as to generation. Third, dialogue coherence is determined by analyzing the content of contributions to the dialogue with the help of a partial logic (Abdallah, 1995): For each new utterance, the dialogue manager computes how it contributes to the completion of one of the currently active plans. Hence, the dialogue model extends the information state approach of (Larsson, 2002). Coherence in our dialogue model is not limited to the expressiveness of finite automata, but is defined as a contribution to a plan — a feature that is not found in other configurable systems such as GALAXY (Seneff et al., 1998).

References

- Nait Abdallah. 1995. *The Logic of Partial Information*. Springer.
- Steven Abney. 1991. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle-based Parsing*. Kluwer, Dordrecht.
- James F. Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, pages 1–8, Santa Fe, NM.
- Nicholas Asher and Anthony Gillies. 2003. Common ground, corrections and coordination. *Argumentation*, 17(4):481–512.
- Astrid Brietzmann and Günther Görz. 1982. Pragmatics in speech understanding – revisited. In J. Horecky, editor, *Proceedings of the Ninth International Conference on Computational Linguistics*, pages 49–54, Amsterdam. North-Holland.
- Kerstin Bücher, Michael Knorr, and Bernd Ludwig. 2002. Anything to clarify? report your parsing ambiguities! In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 465–469.
- B. De Carolis, C. Pelachaud, and I. Poggi. 2000. Verbal and non verbal discourse planning.
- J. Cassell, M. Stone, and H. Yan. 2000. Coordination and context-dependence in the generation of embodied conversation.
- S. Fischer, W. Kießling, S. Holland, and M. Fleder. 2002. The cosima prototype for multi-objective bargaining. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1364–1371, Bologna, Italy.
- Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.
- Jörg Hoffmann and Bernhard Nebel. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.
- Martin Klarner and Bernd Ludwig. 2004. Hybrid natural language generation in a spoken language dialog system. In *Proc. KI-2004*, Ulm.
- L. Lamel, S. Rosset, J. Gauvin, S. Bennacef, and G. Prouts. 1998. The limsi arise system. In *Proceedings of IEEE 4th Workshop on Interactive Voice Technology for Telecommunications Applications*, pages 209–214, Torino, Italy.
- Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Department of Linguistics, Göteborg University, Göteborg, Sweden.
- Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. 1998. Galaxy-ii: A reference architecture for conversational system development. In *Proceedings of IC-SLP 98*, Sydney, Australia.
- Gertjaan van Noord. 1990. An overview of head-driven bottom-up generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, Berlin, Heidelberg, New York.