Making Sense of Partial*

Markus Löckelt and **Tilman Becker** and **Norbert Pfleger** and **Jan Alexandersson**
DFKI GmbH,
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{loeckelt|becker|pfleger|janal}@dfki.de

Abstract

We address the challenging task of processing partial multimodal utterances in a mixed-initiative dialogue system for multiple applications such as information seeking, device control etc. Based on four different types of expectations computed by our action planner (aka dialog manager), we distinguish between expected utterances, various degrees of unexpected but plausible utterances, and uninterpretable utterances. We include a description of the backbone architecture and the representation formalisms used.

1 Introduction

An important characteristic of mixed-initiative dialogue are partial utterances that can only be interpreted in the context of the previous dialogue. The setting in which we investigate the interpretation of partial utterances is the multimodal, mixed-initiative dialogue system SmartKom. However, it is our larger goal to develop a core dialog back-bone. In fact, some of the modules described here have already been used in other projects. In SmartKom, communicative actions include spoken utterances and two-dimensional gestures (pointing, encircling etc.) by the user and also by an animated presentation agent. Since the analyses of both modalities are integrated into a common representation, our processing mechanisms are actually modality-independent. In the following we will use *utterance* to include communicative actions in all modalities.

There are a number of phenomena that occur naturally in such dialogues and systems, including the need for robustness against fairly common recognition errors. In this paper, we focus on partial utterances: Those in the context of user-initiative, usually additions to or changes of the current discourse context and those in the context of user-response, usually a reaction to a system request.

This paper first presents the dialog system SmartKom, its discourse and domain representation language and the data flow of processing. Section 2.2 presents the discourse modeler and 2.3 goes into more detail about planning a system action and supplying expectations for the next user utterance. The central sections are 3 and 4 which describe the details of interpretation as the integration of partial utterances into a coherent dialog context. Before we conclude our paper we discuss and compare our approach in the light of QUD in section 5.

2 Architecture

Our back-bone is divided into different modules (see figure 1), each specialized on a different task. A multi-blackboard architecture (see, e. g., (Wahlster, 2000)) is used for communication where different modules publish or subscribe to indicate read or write permission for so-called data pools. All modules within the back-bone exchange information using a common representation: the domain model (see below). For the analysis part of the back-bone, either a single *application object* or one or more *subobjects* are wrapped into a *hypothesis*, containing additional information, like syntactic information, scores, dialogue acts etc. A sequence of hypotheses forms a *hypotheses sequence* and finally several (alternate) hypotheses sequences form a *hypothesis lattice*.

2.1 Domain Model

The processing of discourse information is not independent of the representation formalism. Thus we will briefly characterize the approach taken in SmartKom, see (Gurevych et al., 2002) and also section 4. The representation is similar to frame-based approaches (Minsky, 1975). It is based on an ontology of application objects and subobjects. Application objects, e. g., a movie_ticket_reservation event, contain subobjects, e. g., movie_theater, show_time, and movie_information. Subobjects are recursively composed of other subobjects, e. g., the movie_theater subobject contains contact information which includes an address which includes a street_name etc. The list of top-level objects, i. e., application objects is of course determined by the set of applications. In SmartKom, there are currently about 10 different application objects. In ad-

* The research within SmartKom presented here is funded by the German Ministry of Research and Technology under grant 01 IL 905.

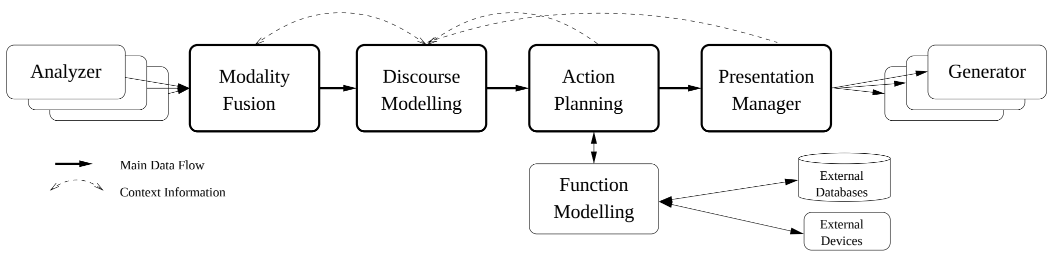

Figure 1: Architecture of the back-bone

dition to an application object, an analyzed utterance will contain a *goal*, e. g., *info* or *reserve* for a movie. However, the objects cannot be combined freely, as in logical frameworks such as those used for natural language semantics. Instead, the entire ontology is predefined for the given applications.

We call the position of a subobject in an application object a *path*. Some paths are called *slots*. These are paths within an application object that are meaningful for the action planner, e. g., the path leading to a (begin) time expression in a movie application object.

Furthermore, the application objects and subobjects are represented as *typed feature structures* with an inheritance *tree*[1] for the types. The type hierarchy is used to express relations – especially shared substructures – between objects.

2.2 Discourse Modelling

In a system where the user communicates with, e. g., spoken language and gestures, the recognition modules produce many hypotheses in addition to the semantic ambiguities arising while analyzing. We have chosen to tackle this challenge by letting all analysis components compute a score for each hypothesis. These scores form the decision base for the selection of a most probable hypothesis sequence.

Modality hypotheses are brought together by the modality fusion module into an intention hypothesis lattice (see (Johnston et al., 1997) for a similar approach) and are passed on to the discourse modeller. Before selecting a hypotheses sequence[2] the discourse modeller performs two tasks: First, it scores (see (Pfleger et al., 2002)) each hypothesis based on how well it fits into the current discourse context (*validation*). Second, appropriate contextual information (see section 3) is added to the hypothesis (*enrichment*). Important here is that although the input hypothesis may contain subobjects representing partial analyses, the output from the discourse modeller should contain only an application object (see also section 2.3). The enrichment is ensured using one simple and robust processing mechanism. In the case of a hypothesis containing an application object, the application object is compared with the discourse context using a non-monotonic operation we call OVERLAY (Alexandersson and Becker, 2001; Pfleger et al., 2002). The overlay operation is based on unification and works on two structures we denote *covering* and *background* where the former represents new and the latter old information. In case of conflicting information, i. e., unification would fail, overlay overwrites the conflicting information in the background. Important is that in case of a type clash, the background is first *assimilated* to the type of the covering thus making it possible to inherit parts of the background despite a type clash. The assimilation is computed via the least upper bound of the covering and the background[3]. If the hypothesis contains one or more subobjects, in a first step the most probable meaning of the subobject(s) is determined. Here, the expectations from the action planner give the expected/possible/filled paths which represent possible interpretations of the subobject(s). These paths are used to construct a new instance of the appropriate application object, an operation we call *bridging*. We call the result of the bridging operation the *interpretation* of the subobject(s). In the next step, the interpretation is overlayed over the appropriate background. During the overlay operation, a score is computed which represents how well the hypothesis fits the context (Pfleger et al., 2002). Finally, the hypothesis with the best overall score is selected and passed to the action planner which plans and triggers actions like questioning a data base or operating on an external device (see section 2.3) and/or appropriate presentations.

In case the subobjects could not be integrated into

[1]Although the implementation in SMARTKOM is based on an inheritance tree, our approach is general enough to handle inheritance lattices.

[2]SMARTKOM features two additional modules for intention selection and, e. g., incorporating default information to the hypotheses. The former may also take other factors into account; however, it is not an integral part of our core backbone.

[3]. . . which might result in no inherited information. THis is the case if the least upper bound is top

an application object, fallback processing is responsible for resolving the meaning of this particular intention.

2.3 Action Planning

As an example consider the following user utterance:

(1) User: I want to record a film on channel two.

It would result in an intention hypothesis containing a goal to be reached in the VCR application, that of recording a film on the VCR, and provide an application object identifying the channel to be recorded.

If a hypothesis sequence arrives, the action planner processes the hypotheses in turn and devises a sequence of steps to establish the goals of the user. In the example task, the system needs some additional information. Minimally, this would be start time and end time. To reach the goal, a sequence of steps – a plan – is devised to accomplish completion of the task. This may involve actions on external functionalities like databases or abstract devices as well as seizing the dialogue initiative to request additional information from the user.

Figure 2 schematically depicts the plan for the goal VCR_record. The boxes indicate plan operators with their pre- and postconditions, as well as the actions and presentations generated during execution. The arrows indicate the (partial) plan execution ordering. To reach the goal, several "tracks" must be followed, each one ultimately establishing conditions needed by the goal. Since the channel was already provided in the utterance, no VCR_getChannel operation needs to be part of this plan.

To execute the plan, its tracks can be followed in arbitrary order. One possibility is to start with the plan operator VCR_getStartTime. The action planner seizes the dialogue initiative and triggers a presentation asking for the start time as specified in the plan operator, then halts since the postcondition does not hold (VCR_startTime is not known). It then waits for a message providing needed information – in this case, new user input in response to the presented request – then tries to proceed with plan execution.

In the example, two pieces of information of type "time specification" are needed to complete the overall task, a start and an end time. If it can be inferred from the form of the user utterance alone whether it is about start or end time, the data can be integrated at the correct position in the application object unambiguously. This is not always the case, though. In the example, an answer giving a time could be about a start or end time, but an end time is not really plausible because the system just asked for a start time. The discourse modeller will try to use discourse context to resolve ambiguities and insert subobjects at the correct positions. The action planner tries to help this resolution by publishing its expectations regarding user input.

2.4 Expectations

An *expectation* is a data structure providing predictions about anticipated user input. When the system has seized the initiative and the user is expected to react, an expectation is published to support the scoring of different interpretation alternatives for the reaction by specifying context information. Most importantly it divides the slots into four classes, depending on the current context:

1. A list of *expected slots*: Typically, application subobjects that the user has just been asked to provide.
2. A list of *possible slots*: application subobjects that are not filled and not focused on by the dialog at the moment, but of which it is in principle possible that they may be filled even if not asked for, e. g., the user may specify a recording time when asked to insert a medium.
3. A list of *filled slots*: subobjects that already have been assigned a value in the course of the dialogue. If the user utterance is about retracting or changing already present information, it should be interpreted with regard to filled slots.
4. All remaining slots implicitly belong to the class *other slots* which cannot be interpreted in the current context and are either misinterpretations or must be considered un-cooperative user behavior.

The expectation contains some additional information like the *type of the application object* that is currently active and/or talked about.

3 Integration of Partial

As discussed in section 2.2, the major tasks of the discourse modeler are validation and enrichment of the user utterance. For partial utterances which are analysed as subobjects, this includes the integration into the preceding context to achieve a coherent discourse.

The general procedure for this, see also section 2.2, is to integrate the subobjects into a new application object of the same type as the object of the focused application. Enrichment is achieved by applying the OVERLAY-operation (see (Alexandersson and Becker, 2001)) to the new application object and the application object in focus. We consider three cases of partial utterances with respect to their interpretability:

1. the system has the initiative and the user responds elliptically to a system request

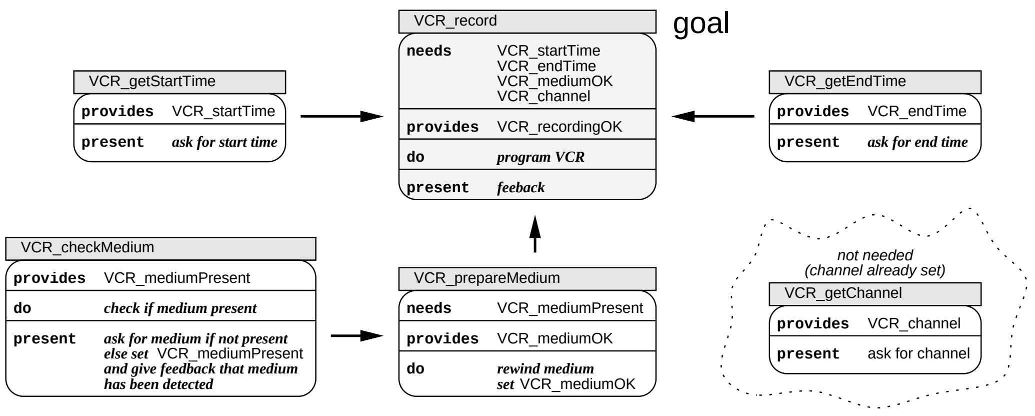
VCR_record
goal
needs VCR_startTime VCR_endTime VCR_mediumOK VCR_channel
provides VCR_recordingOK
do program VCR
present feeback
VCR_getStartTime
provides VCR_startTime
present ask for start time
VCR_getEndTime
provides VCR_endTime
present ask for end time
VCR_checkMedium
provides VCR_mediumPresent
do check if medium present
present ask for medium if not present else set VCR_mediumPresent and give feedback that medium has been detected
VCR_prepareMedium
needs VCR_mediumPresent
provides VCR_mediumOK
do rewind medium set VCR_mediumOK
not needed (channel already set)
VCR_getChannel
provides VCR_channel
present ask for channel

Figure 2: A schematic plan for *"I want to record a film on channel two"*

2. an elliptical user utterance does not correspond immediately to a request but can be interpreted in context
3. an elliptical user utterance cannot be interpreted, caused either by misinterpretation by the analysis module or non-cooperative user behavior.

In case 1 the answer uttered by the user contains the requested subobject (i. e. the user skips the constituents that are already mentioned in the question). The identification of an elliptical answer to the system request is based on the presence of an *expected slot* of the same type as the incoming subobject. Additionally, it is possible that the user is providing other relevant information, as in case 2. This can be interpreted via *possible* or *filled slot*s. Section 3.2 describes the processing of user responses and an example is given in section 4.1.

In case 2 the user might provide unasked-for information that still fits into the discourse history, e. g. a restriction of "only channel two" upon presentation of an overview over the current TV program. Another possibility is that the user intends to change the value of an already set slot of an application object by uttering only the changed subobject, e. g., "and tonight?" in the same context.

The correct interpretation of such a subobject is a replication of the request from context including the new or changed subobject. This requires the presence of *possible* or *filled slot*s. Either there are no *expected slot*s or the subobject is incompatible with all of them. Section 3.3 describes the identification and processing of such partial utterances and in section 4.2 an example is given.

Case 3 is met when the discourse modeller is not able to integrate the subobject into an application object; this might occur in case of non-cooperative user behaviour or recognition errors. In such cases, supplemental recovery strategies are used.

Given these three classes of partial utterances and the four classes of slots as defined for our expectations, we can summarize the possible combinations (of which more than one can occur in the same utterance) in the table 1.

	User Initiative	System Initiative
Expected slot	NA	expected, unify, plan continues
Possible slot	plausible, unify, replanning	plausible, unify, replanning
Filled slot	plausible, overlay, replanning	plausible, overlay, replanning
Other	implausible – recover strategies	

Table 1: Possible combinations of partial utterances and different expectations given user or system initiative.

Note that if the user retains the initiative, no expected slots are available as marked by NA in the upper left cell. If the initiative remains with the system and the user supplies an expected answer (upper right cell), the action plan continues as in a simple system-initiative dialog system.

The processing of possible and filled slots does not depend on the status of initiative. Also, both are handled by a single mechanism, using the OVERLAY-operation. However, in the case of possible slots, OVERLAYreduces to pure unification. The special properties of overwriting are only used for already filled slots. These differences are reflected in different scores, see (Pfleger et al., 2002). The (re-)setting of filled and possible slots can give rise to the need to adapt the current plan, reflecting the mixed-initiative situation.

To illustrate the process of integrating a partial user utterance in the context preceding it, we now take a brief look at the underlying structures and operations of the discourse modeller.

3.1 Context Representation

Our approach to context representation (Pfleger, 2002) is based on a three-tiered context representation (LuperFoy, 1991) and mental representation as in (Salmon-Alt, 2000).

For the work presented here, we focus on the three layers of the context representation. In the discourse layer, a *discourse object* (DO) represents a concept which can be referred to. It is introduced into the discourse either by means of language (user and system) or graphics (system). A DO keeps information about each mentioning depending on the manner of presentation. Additionally, each DO has access to its corresponding application object or subobject. Access to DOs in the discourse layer is restricted by a local focus stack. In the *Modality Layer* there are three types of objects: *Linguistic Objects* (LO), *Gesture Objects* (GO), and *Visual Objects* (VO). The third layer consists of *domain objects* are instances of our domain model, including, e. g., the application objects. Figure 3 shows how the three-tiered context representation and the local focus stack are connected. For each turn there exists exactly one application object or at least one subobject representing the current user or system intention. These objects are stored in order of occurrence in the domain layer and provide the domain information for the lower layers.

3.2 Processing User Responses

In the case of a user response to a system request, the integration of subobjects is guided by the list of *expected slot*s which determine the paths of the expected slot fillers within the current application object. For the integration of the subobjects, first, a new application object of the same type as the current application is constructed. For each subobject in the hypotheses its type is compared with the types of the *expected slot*s. In the case of a match, the subobject is integrated into the new application object using the path information of the matching slot. Then, the newly created and extended application object is OVERLAY-ed over the focused application object for enrichment with previous discourse information. If some subobjects do not match with any of the expected slots, the integration continues as described in the following section.

3.3 Processing Plausible Partial Utterances

In general, *possible slot*s are processed just like *expected slot*s. If the user tries to change a *filled slot*, this is implemented by searching for a discourse object of the same type in the local focus stack. Such a matching discourse object will always be found since the slot had been filled at some point in history. The scoring, however, might be low, e. g., in case it was mentioned a long time ago. A new application object of the same type as the current application is constructed. The subobject is integrated into this application object using the path information of the matching discourse object. Finally, the new application object is OVERLAY-ed over the application object of the previous user turn.

4 Two Examples

The first, shorter example shows the distinctions between expected and possible slots. The more elaborate second example, dealing with a filled slot, gives more details of the context representation.

4.1 Programming a VCR

On the basis of the following example we demonstrate the process of integrating a subobject in the case of system-initiative with a matching *expected slot*:

(2) User: I want to record a film.

(3) System: When should I start recording?

(4) User: From 1:30pm on channel two.

Analysis of (2) produces an intention hypothesis containing the setting of a goal VCR_record and an empty application object for the VCR application. The action planner accepts VCR_record as the current goal and constructs a plan to acquire the necessary data for a recording.

In this case, it decides that the time slot, VCR_startTime, should be filled first, and the plan operator that provides this slot takes the discourse initiative by triggering presentation (3) that asks for it. An expectation is published that contains VCR as the type of the current application object, VCR_startTime as the (only) expected slot, no filled slots (save maybe for possible application defaults), and, some possible slots, e. g., VCR_endTime and VCR_channel.

When the user answers (3) by (4), an intention containing two subobjects is passed from the modality fusion to the discourse modeller, which then uses the expectation to determine how to integrate the new information into a single application object. In (4), the expected slot is filled by *"1:30pm,"* but also one of the possible slots by *"channel two."* The two subobjects are integrated into a new application object which is afterwards OVERLAY-ed over the current application object stored from the previous turn. The resulting application object is passed on to the action planner, which will store the slot settings and continue the dialogue to reach the goal of programming the VCR.

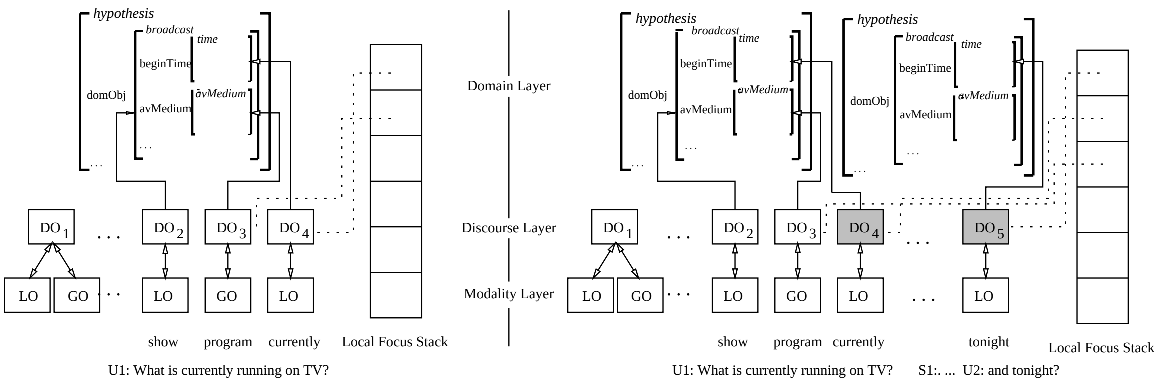

Figure 3: Two sample configurations of the context representation

4.2 Watching TV

With this example we demonstrate in some more detail the process of integrating a subobject in the case of user-initiative with no matching *expected-slot*:

(5) User: What is currently running on TV?

(6) System: The following (SMARTAKUS points at a list of programs) programs are currently running.

(7) User: And tonight?

The left part of figure 3 shows an excerpt of the configuration of the context representation after (5) has been processed. For example, the discourse object DO_4, introduced by the user's uttering of *"currently,"* points to the slot TV_beginTime in the current application object stored in the domain layer. It is also accessible through the focus stack.

The right part of figure 3 shows the situation after (7) has been analyzed. DO_5 is introduced by the user's utterance of *"tonight."* There is no *expected slot* and DO_5 does not match any of the *possible slot*s. This triggers a search in the local focus stack for a discourse object corresponding to a *filled slot*. During this search, DO_4 is found. Then, a new application object containing the subobject of the user utterance (7) under the path for the filled slot (TV_beginTime) is created and enriched with the remaining information of the application object of (5), using the OVERLAY-operation. Finally, the resulting application object is passed on.

5 Discussion

We characterize our approach in the light of (Ginzburg, 1996). In (Ginzburg, 1996), his *Dialogue Score Board* (DSB) keeps track of:

- The set of currently accepted FACTS.
- The syntax and semantics of the LATEST-MOVE.
- QUD: A partially ordered repository that specifies the currently discussable questions.

Although Ginzburg's work on QUD aims for a more general framework than ours, there are some similarities between our approaches. Our notion of filled slots seems to correspond the the facts in the DSB. There is a close relation between the QUD and our exptected and maybe also our possible slots.

Amongst the differences between our approaches we have meta-talk. Instead of allowing for meta-talk we specialize on narrow but multiple domains and tasks, such as information search, device control, etc. In these scenarios, however, we consider our entire context for resolution of partial utterances. Another difference is the absence of traditional semantics in our approach. Instead we use an ontology-based domain model.

The salient feature of our frame work comes out clear in table 1. There we show a fine grained characterization of possible short utterances (which we call partial utterances) in different situations and their corresponding consequences. For all different cases, we use one robust and simple mechanism which treats all different expectations (expected, possible and filled slots): OVERLAY. Our scoring function distinguishes the different exptectations in such a way that unifiability is rewarded whereas overwriting is not (see (Pfleger et al., 2002; Pfleger, 2002) for more details).

6 Conclusion

We have shown how partial (multimodal) utterances can be interpreted by integrating them into an appropriate dialog context. While our approach hinges on a frame-like representation of dialog context and corresponding processing operations, some of the details have proven very helpful but are not essential to the core approach, e. g., the use a three-tiered discourse history, typed feature structures and the OVERLAY operation.

We have shown how in such an approach action planning can provide expectations to make sense of partial utterances in various situations. Especially for the interpretation unexpected (partial) utterances, a complex discourse history and focus stack are necessary to find the most plausible context for their integration. The approach is general enough to handle system expectations with the same processing mechanism.

References

Jan Alexandersson and Tilman Becker. 2001. Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System. In *Workshop Notes of the IJCAI-01 Workshop on "Knowledge and Reasoning in Practical Dialogue Systems"*, Seattle, Washington, August. `http://www.ida.liu.se/labs/nlplab/ijcai-ws-01/alex.pdf`.

Christian Ebert, Shalom Lappin, Howard Gregory, and Nicolas Nicolov. 2001. Generating full paraphrases of fragments in a dialogue interpretation system. In *Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue*, Aalborg, Denmark, September, 1-2.

Jonathan Ginzburg. 1996. Dynamics and the semantics of dialogue. In J. Seligman, editor, *Language, Logic and Computation*, volume 1. CSLI Lecture NOTES, CSLI, Stanford.

Iryna Gurevych, Robert Porzel, and Michael Strube. 2002. Annotating the Semantic Consistency of Speech Recognition Hypotheses. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, pages 46–49, Philadelphia, PA, July. Association for Computational Linguistics.

Michael Johnston, Philip. R. Cohen, David McGee, Sharon L. Oviatt, James A. Pittman, and Ira Smith. 1997. Unification based multimodal integration. In *Proceedings of the 35th ACL*, pages 281–288, Madrid, Spain.

Susann LuperFoy. 1991. *Discourse Pegs: A Computational Analysis of Context-Dependent Referring Expressions.* Ph.D. thesis, University of Texas at Austin, December.

Marvin Minsky. 1975. A framework for representing knowledge. In Patrick Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw Hill, New York.

Norbert Pfleger, Jan Alexandersson, and Tilman Becker. 2002. Scoring functions for overlay and their application in discourse processing. In *KONVENS-02*, Saarbrücken, September – October.

Norbert Pfleger. 2002. Discourse processing in multimodal dialogues. Master's thesis, Unversität des Saarlandes. Forthcoming.

Susanne Salmon-Alt. 2000. Interpreting referring expressions by restructuring context. In *Proceedings of ESSLLI 2000*, Birmingham, UK. Student Session.

Wolfgang Wahlster, editor. 2000. *VERBMOBIL: Foundations of Speech-to-Speech Translation.* Springer.