Anything to Clarify? Report your Parsing Ambiguities!

Kerstin Bücher[1] and **Michael Knorr**[1] and **Bernd Ludwig**[2]

Abstract. An important factor for the acceptance of spoken dialogue systems is their ability to react flexibly on misunderstandings between user and system. This paper addresses the issue of grounding utterances in task-oriented human-computer dialogues. It focuses on the aspect of handling ambiguities while parsing word lattices from a speech recognizer: The parser detects the origin and the type of ambiguities which are reported to the dialogue manager as comments to the list of readings for the user's utterance. This way, disambiguation is delegated to the dialogue manager and accomplished either by exploiting the application situation or by initiating clarification dialogues that are suitable for the dialogue situation.

1 INTRODUCTION

Experience with spoken dialogue systems such as EVAR [9] and in-depth analysis of dialogues conducted with this system has shown that one of the main reasons for failed human-computer interactions is the system's difficulty to integrate new utterances in the dialogue context (see [4]). A detailed analysis of the reasons why interactions failed shows that there are two dimensions of integration to be considered. First, how are phrases related to each other within an utterance? This issue becomes particularly important when misrecognitions by the speech recognizer created an output that is syntactically ill formed with respect to a given formal grammar. Keyword spotting approaches, often considered as a robust way of syntactic analysis, are unable to analyze the lack of meaning of the whole utterance. The disparity between the user's intention and the constructed meaning for an utterance has shown to be a main source of misunderstanding in human-computer interaction. Second, the meaning of an utterance has to be integrated in the dialogue context. Normally, implementations of dialogue managers rely blindly on the results computed by speech recognizer and parser. However, human-human dialogues show that persons in a conversation often reassure themselves whether they are understanding the speaker. So, humans manage to perform dialogues not only on the task to be carried out in an interaction, but also on the issue of reception.

Rethinking the interaction between parser and dialogue system on the basis of the experiences outlined so far, we present an approach that combines closely the processes of parsing an utterance and its integration in a dialogue. In order to improve the cooperation between the parser and the dialogue manager, we completely reorganize the classical chart parsing approach by splitting parsing into two phases. In the first phase, a chart parser segments input into chunks; in the second phase, the dependency relations among the chunks (see section 2) are analyzed. In four steps, syntactic, semantic, and pragmatic constraints are applied in a bottom-up traversal to test the viability of dependencies. If possible, Discourse Representation Structures (DRS) as defined in [10] are composed and rated using a combination of scores which take into account the coverage of the interpretation, the acoustic quality, the dialogue context and which valencies of the chunks are filled (see section 3).

When the parser discovers an ambiguity it creates an ambiguity report (see section 4) for the dialogue manager which might be able to disambiguate the utterance using the dialogue context.

The dialogue manager follows an approach to rational interaction that reacts flexibly to varying situations depending on the input it receives. From the parser, it obtains as input DRSs and ambiguity reports for the differences between the DRSs. If the dialogue manager fails to select a single DRS in order to integrate the parsed utterance in the dialogue context, it uses the content of the ambiguity report to initiate a clarification dialogue (see section 5).

The parser and the dialogue manager are implemented and used in the EMBASSI[3] project providing multi-modal assistance for controlling audio and video equipment. The lexicon contains about 800 stems.

This paper is structured as follows: In section 2 we explain the parsing process, in section 3 and section 4 we present the implementation of the parser; finally, section 5 shows how the parsing results are integrated in the dialogue situation.

2 THO-PHASE PARSING

Building a grammar for the parsing of spoken German has two challenges: First, the parser has to be able to process input that is ungrammatical or incomplete, since incomplete or interrupted utterances, self-corrections and repairs, etc. are common in spontaneous speech. Additionally, given the error rates of speech recognizers, even with correct input the speech recognizer may produce an output which is not grammatical, and the parser has to cope with that. Second, German is a language with fairly free word order, also allowing for discontinuous constituents. Therefore, the grammar cannot rely only on linear sequence as its main concept. We try to overcome these problems by designing a two-phase parsing process.

2.1 The First Phase

The first phase works with a grammar that employs phrase structure rules to build small phrases, called chunks (this approach is similar to the one of [1]). A chunk consists of a head element and another constituent that is a possible filler of a free position in the head's (X-Bar-) structure. The filler usually is a specifier (the determiner in case of a noun phrase) or a modifier (e.g. an adjective phrase modifying

[1] Computer Science Institute (INF 8), University of Erlangen-Nuremberg, Germany

[2] FORWISS (Research Group for Knowledge Processing), University of Erlangen-Nuremberg, Germany

[3] The research presented in this paper has been carried out in the framework of the EMBASSI project. (Grant Nr.: 01IL9904F8).

a noun). A chunk may also consist only of its head. Whether two elements can be combined is constrained by their feature structure which is used for unification, and by linear order requirements: Only elements that are adjacent to each other and that occur in a fixed order can be connected to form a chunk.

Let's consider an example. The utterance "Den Krimi um acht aufnehmen" ("Record the detective story at eight") is analyzed as a sequence of three chunks:

- the DP "den Krimi" built by conjoining the determiner "den" (the syntactic head) and the NP "Krimi",
- the PP "um acht" and
- the VP "aufnehmen"

To build the DP "den Krimi" we need two grammar rules:

NP:	N:	DP:	DET NP:
	head = N:		head = DET:
	NP = N:		DET agreement = NP agreement,
			DP = NP:

Chunks are not only syntactically correct units but are semantically well-formed, too. At this early stage of parsing we already have means to interpret the user's utterance. In the lexicon, each element has in its entry information about its morphological features (lexical category, inflection features, gender, case etc.) and about its semantics, represented as a DRS. When combining two elements, the parser checks the compatibility of the morphological features (e.g. agreement in case of the combination of a determiner with an NP) and merges their DRSs resulting in a DRS for the chunk. This way, each chunk gets an interpretation. The meaning of the utterance is composed by finding the relation between the chunks and their interpretations. This is put off to phase 2 of the parsing process. In case of an ill-formed input, the utterance can be at least partially interpreted. In section 3 we present how the best interpretation is selected.

In the first parsing phase, only those elements are combined that occur in a fixed order: while "den Krimi" is perfectly well-formed, the reverse order "Krimi den" is not. In contrast, the chunks themselves are not restricted to a certain position within the sentence: Either sequence, "Um acht den Krimi aufnehmen" and "Den Krimi um acht aufnehmen", is grammatical. The first parsing step also ignores the fact that the PP "um acht" actually may be attached to either the DP "den Krimi" or the VP. These issues are addressed in phase two of the parsing process.

2.2 The Second Phase

Phase 2 of the parsing process relies on a kind of dependency grammar that for each chunk of phase 1 gives a list of possible syntactic functions the chunk may have. The options are constrained by the morphological features of the chunk, e.g. an NP chunk can function as subject only if its case feature has nominative as its value.

Then, the valencies of each chunk are filled by combining it with other chunks, e.g. building a verb phrase from a verb and its direct object. Again, these combinations are gated by syntactic and semantic constraints. Informations about the valencies of a word are stored in the case frame of that word. The term *valency* here is used in a broader sense: it includes not only obligatory elements needed to make a phrase syntactically complete; more than that, the case frames list all semantically and pragmatically suitable modifications and their syntactic representations, e.g. attributes for nouns or adverbials for verbs. The suitability of the modification is determined by the application ontology.

In contrast to most approaches, not only verbs have case frames. We also need to pragmatically connect for example the DP "den Krimi" with its modifying attribute, the PP "um acht". The case frame for a specific chunk is selected by the semantic head of that chunk: In the case of the DP "den Krimi" it is not the syntactic head DET but the content word "Krimi" whose case frame is to be used. The semantic head is specified in the chunk building rule of phase 1 (here: DP = NP, i.e. the DP inherits all properties including the case frame of the NP). The attachment of the PP to the DP here is guided by the entry for "Krimi" in the case frame lexicon that allows for modification by a chunk whose semantic interpretation is pragmatically suitable, e.g. a starting time. The verb "aufnehmen" has an entry, too, that fits the semantics of the PP, i.e. an adverbial with the semantic interpretation of a starting time, specifying the starting point of the action.

How this kind of parsing ambiguities are handled is the topic of the remainder of the paper. The result of parsing phase 2, however, gives at least one interpretation of the whole utterance. A characteristic property of this second phase is that we control the chunk-combination process at four linguistic levels: Each combination is checked for its morphologic, syntactic, semantic, and pragmatic correctness. In case of different results, e.g. bad morphology (mostly through recognition errors) but perfect pragmatics, pragmatics wins. In case of an ambiguity, well, just go on reading.

3 IMPLEMENTATION OF THE PARSER

As explained before, the second phase of the parser uses dependency grammar to find relations between the chunks built in the first phase. It then traverses the dependency tree to built readings for the utterance. Finally the most probable interpretations are selected. This phase can be divided into four parts:

1. Building an agenda of possible dependencies
2. Testing the syntactic and semantic viability of the dependencies
3. Traversing the dependency tree and assembling interpretations
4. Selecting the best interpretations

3.1 Checking the Dependencies

In the first part, the parser constructs an agenda which contains all possible dependencies from a chunk to all other chunks. For each chunk that has a case frame the parser takes all dependencies for its syntactic category from the dependency file. If, for example, the chunk is a verbal phrase, the parser finds the dependencies `subj`, `dirobj` and `adverbial`. For all other chunks that do not overlap with this chunk it creates an agenda element for each dependency.

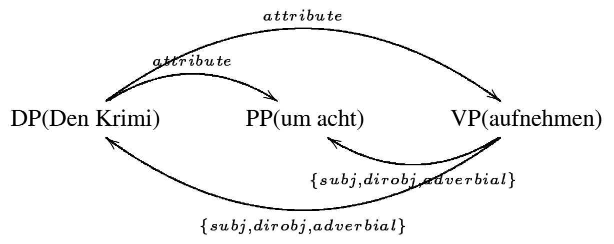

Figure 1. Dependencies for "Den Krimi um acht aufnehmen".

Figure 1 shows the eight dependencies put on the agenda for the example. In the next part, the syntactic and semantic constraints for each dependency on the agenda are checked. The syntactic check tests, if the dependent chunk has one of the categories that can satisfy the dependency and if the constraints on the features of dependent and regent are met. In the example only three dependencies pass this test: The PP could be an attribute to the DP or an adverbial to the VP and the DP is the direct object of the VP. For the semantic check the parser goes through all syntactically suitable slots of the regent's case frame and checks whether the dependent chunk meets the semantic and pragmatic requirements of a slot. If more than one slot matches an ambiguity report is generated, as explained in section 4. In the example, all remaining dependencies pass the semantic test.

3.2 Assembling Interpretations

When the parser has checked all possible dependencies, it traverses the dependency graph in a bottom-up fashion to build the possible interpretations of the utterance. When the parser has run through all of the chunks, each chunk contains a set of readings. In our example "Den Krimi um acht aufnehmen" the parser adds no readings for the chunk "um acht", which is at the bottom of the dependency tree. It constructs two readings for "den Krimi", namely the original meaning and the combination with the prepositional phrase, and it builds five interpretations for "aufnehmen".

3.3 Selecting the Best Interpretations

Finally the parser selects the readings for further processing. Consequently, it first checks all DRSs. If a reading is subsumed by another which has a better score, the former is deleted. The remaining DRSs are passed to the dialogue manager. The score of a chart edge or reading is calculated from a number of specific scores:

- An acoustic score obtained from the speech recognizer
- The length of the utterance part spanned by the reading
- The dialogue context. When the dialogue manager asks a question, it sends the parser a list of the concepts of the expected answers. Readings containing one of these are scored better.
- Each valency has an associated score, which is higher for essential slots than for facultative ones.

In some cases it will not be possible to construct a reading that spans the whole utterance. Then the best fragments will be handed to the dialogue manager.

3.4 Packed DRS

In the second phase of the parser, the semantic representations for the different readings of the input are built. To do this the semantic representations of the chunks are combined to larger and larger representations. Unfortunately, in the case of ambiguities, we can not simply throw away a representation once it is used in a larger representation, since it still may be needed to build a different reading. Because of this we get many representations all made up from the same basic parts.

Therefore, we decided that instead of actually constructing the DRSs we just create "building plans". Additionally we keep a pool of DRS segments. These segments have disjunctive nodes at which other segments can be plugged in. Apart from that, the disjunctive nodes are used like discourse referents. At the end of the parse, the DRS for the best interpretations are created from the segments according to plugging functions (the "building plans"). This process starts with looking up the root segment under the root node Δ_0. Then the disjunctive nodes of the root segment are filled with other segments according to the plugging function. If no value is defined for a disjunctive node, the condition in which it appears is removed. This is repeated for each segment plugged in, until every disjunctive node is either filled or removed.

$$S_1 = \begin{array}{|l|}\hline \text{x, y, } \Delta_1 \\ \hline \text{AvEvent(x, y)} \\ \text{has-genre(y, "Krimi")} \\ \text{Genre("Krimi")} \\ \text{has-timeinterval(x, } \Delta_1\text{)} \\ \hline \end{array}$$

$$S_2 = \begin{array}{|l|}\hline \text{x, y} \\ \hline \text{TimeInterval(x)} \\ \text{has-starttime(x, y)} \\ \text{clocktime(y)} \\ \text{has-hour(y, 8)} \\ \hline \end{array}$$

$$S_3 = \begin{array}{|l|}\hline \text{x, } \Delta_2\text{, } \Delta_3 \\ \hline \text{Record(x)} \\ \text{has-avevent(x, } \Delta_2\text{)} \\ \text{has-timeinterval(x, } \Delta_3\text{)} \\ \hline \end{array}$$

$$P_1 = \begin{bmatrix} \Delta_0 & \rightarrow & S_3 \\ \Delta_1 & \rightarrow & S_2 \\ \Delta_2 & \rightarrow & S_1 \end{bmatrix} \qquad P_2 = \begin{bmatrix} \Delta_0 & \rightarrow & S_3 \\ \Delta_2 & \rightarrow & S_1 \\ \Delta_3 & \rightarrow & S_2 \end{bmatrix}$$

Figure 2. DRS pool and plugging functions for "Den Krimi um acht aufnehmen"

Figure 2 shows the slightly simplified DRS pool for the example "Den Krimi um acht aufnehmen", containing a DRS segment for each of the three chunks. On the right it shows the plugging functions for the two best interpretations. P_1 describes the reading in which the film starts at eight, P_2 the reading in which recording starts at eight. During parsing a number of plugging functions are constructed which only cover a part of the utterance. In the example all segments are used in the plugging functions. This is not always the case. If a lexical ambiguity exists in an utterance, there will usually be a segment for each reading of the ambiguous word, while only one of these readings can be used in a plugging function.

In recent years a number of under-specified representations have been developed to tackle the problem of scope ambiguities (see [11] for an introduction). Superficially, our approach is quite similar to the unplugged DRS in Johan Bos' hole semantics framework ([6]). However, there are several important differences: In our unpacking algorithm conditions are removed when their disjunctive nodes are not filled; in other under-specified representations all disjunctive nodes have to be filled, so this situation can not arise. Other representations usually require that all segments are used in every plugging function. This makes it impossible to represent lexical ambiguities and difficult to represent attachment ambiguities. On the other hand, this restriction allows other representations to use simple constraints to specify the possible pluggings, while we currently use lists of plugging functions. We hope to find a set of constraints suitable for defining our plugging functions in the future.

4 AMBIGUITY REPORTS

While many ambiguities can be resolved by the parser, some will remain and have to be handed over to the dialogue manager, which might be able to resolve further ambiguities, for example, by using dialogue context or user preferences. However in some cases it will be necessary to ask the user about the meaning of his utterance.

The parsing result is a list of scored readings for the utterance. In many systems such a list is simply passed to the dialogue manager, which is left to its own devices to figure out the differences between the readings. In our framework, however, the parser informs the dialogue manager about the differences. If it can not resolve the ambiguity itself, it can at least use this information to ask the user a well directed clarification question.

Therefore, the parser marks all ambiguities when they arise, so they can be traced through the parse. It also creates an ambiguity report, which is sent to the dialogue manager, if the ambiguity can not be resolved during parsing. This ambiguity report contains a discourse representation structure that describes the type of the ambiguity and its variants. The readings that follow from using a variant are specified with the `affects-reading` condition. The meaning in which the variants differ is recorded under `has-content`. Depending on the point were it arises, an ambiguity is classified as belonging to one of four types: lexical-ambiguity, pragmatic-ambiguity, attachment-ambiguity or filler-ambiguity.

Whenever entries with different concepts for a word are retrieved from the lexicon the parser classifies this as *lexical ambiguity*. A common example of lexical ambiguities in our domain are names of songs that double as band names. “Fiddler’s Green” for example is an Irish folk song which was performed and recorded by numerous bands, but is also the name of an Irish speed folk group.

Occasionally, words can be used in different pragmatic contexts. This is labeled *pragmatic ambiguity*, an example would be the sentence “I love detective stories”. In the application’s domain model the word “love” can be mapped to the concepts `GiveFavouriteAvEvent`, `GiveFavouriteAvEventLocation` or `GiveFavouriteGenre` which take a TV program, TV station or a genre as second argument.

r_1, r_2, u, v, w, x, x_0, x_1, y
attachment-ambiguity(x)
has-variant(x,x_0)   ambiguity-variant(x_0)
affects-reading(x_0, r_1)   has-content(x_0, y)
AvEvent(y)   has-genre(y, “Krimi”)   Genre(“Krimi”)
has-variant(x,x_1)   ambiguity-variant(x_1)
affects-reading(x_1, r_2)   has-content(x_1, w)
Record(w)   has-pp(x, v)   TimeInterval(v)
has-starttime(v, u)   clocktime(u)   has-hour(u, 8)

Figure 3. Ambiguity report for “Krimi um acht aufnehmen”

The example “Den Krimi um acht aufnehmen” features an *attachment ambiguity*. The parser realizes this when it finds two valid dependencies with “um acht” as dependent. At this point it generates the ambiguity report shown in figure 3. The DRSs of the possible attachment points are stored in `has-content`. Additionally, the role `has-pp` is used to store the pragmatic extension of the phrase that caused the ambiguity.

Filler ambiguities occur if two chunks compete for the same case-frame slot of another chunk, as in the example *“I want to watch the thriller at eight, not at nine”*. Both PP chunks are related to the `has-starttime` slot of “thriller”.

5 MODELLING RATIONAL DIALOGUES

In [3, 7, 8] it is argued that dialogue as a form of rational interaction is a means of cooperatively executing tasks for joint purposes. Allwood ([2]) adds that any dialogue consists of successive communicative contributions. They are actions in a multi-layered plan of interaction fulfilling certain communicative functions. The effect of such actions is an update of the current belief structures of the dialogue participants. So, dialogue interpretation gets linked with cognitive modelling ([5]).

5.1 Updates of the Dialogue Situation

The approach to dialogue analysis presented here separates the dialogue situation as the current state of the interaction from the application situation as the current state of application pragmatics: Utterances evoke actions in the application situation. This way, they change the application’s state; the effects of these actions motivate new speech acts to keep the interaction going until the joint purpose has been reached. Now, under which conditions can an utterance have such an effect on the application? Roughly speaking, if one of its readings can be integrated in the plan for the current joint purpose. As suggested in section 4, there are situations in which output from the parser is ambiguous for the dialogue manager and the content of its belief structure does not suffice for disambiguation. In such a case, an update of the dialogue situation is impossible as the dialogue manager fails to uniquely integrate the utterance in the dialogue situation. Therefore, it is unable to carry out the current task.

5.2 Micro-conversational Events

However, the dialogue situation has been fed by the parser with information about the reasons for the ambiguity and the possible readings. The ambiguity report in figure 3 is incorporated in the dialogue situation depicted in figure 4.

Including syntactic information in dialogue situations (as also proposed in [12]) is motivated by the necessity to keep a dialogue coherent if syntactic or semantic ambiguities of new utterances cannot be resolved. Coherence is accomplished by focusing on the failed preconditions for a unique integration of the utterance.

The importance of representing information about micro-conversational events becomes clear when considering the options of a dialogue system for a clarification dialogue that is not using any

r_1, r_2, u, v, w, x, x_0, x_1, y, s_0, **U**_1
attachment-ambiguity(x)
has-variant(x,x_0)   ambiguity-variant(x_0)
affects-reading(x_0, r_1)   has-content(x_0, y)
AvEvent(y)   has-genre(y, “Krimi”)   Genre(“Krimi”)
has-variant(x,x_1)   ambiguity-variant(x_1)
affects-reading(x_1, r_2)   has-content(x_1, w)
Record(w)
has-pp(x, v)
TimeInterval(v)
has-starttime(v, u)   clocktime(u)   has-hour(u, 8)
situation(s_0)
has-event(s_0,**U**_1)   request(**U**_1)
has-reading(**U**_1,r_1) has-ambiguity(r_1,x)
has-reading(**U**_1,r_2) has-ambiguity(r_1,x)
r_1: ...
r_2: ...

Figure 4. Dialogue situation for an attachment ambiguity

u, v, y
AvEvent(y)   has-genre(y, “Krimi”)   Genre(“Krimi”)
has-timeinterval(y, v)   TimeInterval(v)
has-starttime(v, u)   clocktime(u)   has-hour(u, 8)

u, v, w
Record(w)
has-recordingtime(w, v)
TimeInterval(v)   has-starttime(v, u)
clocktime(u)   has-hour(u, 8)

Figure 5. Readings resulting from the attachment ambiguity

ambiguity report: Assuming the availability of a deep text generation, how could the difference between the two readings be marked? Probably not at all – so, a clarification would not be of much use. We see the ambiguity report as a prerequisite for generating system utterances that are plausible to the user. Understanding the need for clarification, the user is able to answer in a way that really helps disambiguating. Finally, we note that with template-based generation, such clarification dialogues seem impossible to be conducted.

5.3 Grounding Ambiguous Readings

Due to the intended purpose of an utterance in a rational dialogue, several orthogonal aspects of constructing meaning, integrating the utterance in the current discourse situation and relating it to the application situation have to be considered:

- Acoustics: How is the quality of speech recognition?
- Syntax: What is the state and the result of the syntactic analysis?
- Semantics: What is the result of the semantic composition?
- Speech act: What is the effect on the discourse situation?
- Coherence: Does the utterance refer to the current focus?
- Plan: Is there a plan to satisfy the user's intention?
- Action: What is currently done to execute the plan?
- Status: What is the state of the plan execution?

Ambiguity reports affect the syntactic and semantic aspect: they deliver information that no unique meaning could be constructed.The dialogue manager knows that there are two chunks involved in the ambiguity: C_1 and C_2. In order to ground the utterance, it computes the critical parts of the readings r_1 and r_2 (see figure 5).

Using knowledge about the discourse situation the dialogue manager has to decide whether the found ambiguities are relevant. Therefore, it tries to relate the head of each DRS (y or w, respectively) to some already existing discourse referent. If this is successful the conditions in the DRS are used to verify whether it actually refers to the assumed antecedent. If this analysis does not result in a disambiguation, the dialogue manager verifies whether the content of both DRSs is satisfiable in the current application situation hopefully excluding one reading.

5.4 Generating Content for Clarification Dialogues

If all tries to disambiguate the meaning of the utterance fail, the dialogue manager uses the ambiguity report to deliver the content for a clarification question. The intention behind the question is to involve the user in the process of disambiguation, when he selects one of the readings. For the purpose of transparency, the dialogue manager can give the user additional information why it asks this question. Without an ambiguity report, the user could at best be told that his utterance was ambiguous without any explanation. Knowing that the PP in the utterance could not be attached uniquely makes it possible to generate an utterance like *“I am not sure if you talk about the detective story at 8 or if I should start recording at eight.”*

The main benefit of this approach is that handling of parsing ambiguities can be solved in the same way as it is done for pragmatic ambiguities: Like any application component, the parser is not considered a black box which is using only its own sources of information to compute an interpretation for an utterance. It rather works interactively with other components that have access to additional information about the dialogue situation. We think that this approach resembles better the way in which two humans perform clarifications.

6 CONCLUSION

The paper shows an approach to computing information about micro-conversational events during parsing word lattices. This information reports on ambiguities found by lexical, syntactic and semantic analysis in a two stage parsing process. In this way, the parser is able to “explain” to the dialogue manager why it produced several readings for an utterance. The dialogue manager benefits from this additional input as appropriate system turns can make the user aware of the system's reception problems. The presented parser and dialogue manager have been implemented. Experiments with “naive” users have been conducted and are currently being evaluated. If possible, we want to go even further in the sketched direction and rethink the interface and cooperation between parsers and speech recognizers.

REFERENCES

[1] Steven Abney, ‘Parsing by chunks’, in *Principle-based Parsing*, eds., R. Berwick, S. Abney, and C. Tenny, Kluwer, Dordrecht, (1991).

[2] Jens Allwood, ‘Obligations and options in dialogue’, *Think*, **3**, 9–18, (May 1994).

[3] Jens Allwood, ‘Dialog as collective thinking’, in *Brain, Mind, and Physics*, eds., P. Pylkko P. Pylkkanen and H. Hautamaki, volume 33 of *Frontiers in Artificial Intelligence and Applications*, Amsterdam, (1997). IOS Press.

[4] Maria Aretoulaki and Bernd Ludwig, ‘Automaton-descriptions and theorem-proving: A marriage made in heaven?’, *Linköping Electronic Articles in Computer and Information Science*, **4**(22), http://www.ep.liu.se/ea/cis/1999/022/, (1999).

[5] Nicholas Asher and Alex Lascarides, ‘Intentions and information in discourse’, in *Proceedings of the 32nd Annual Meeting of the Association of Computational Linguistics*, pp. 34–41, Las Cruces, USA, (June 1994).

[6] J. Bos, ‘Predicate logic unplugged’, in *Proceedings of the 10th Amsterdam Colloquium*, eds., P. Dekker and M. Stokhof, pp. 133–143. ILLC, University of Amsterdam, (1995).

[7] Sandra Carberry and L. Lambert, ‘A process model for recognizing communicative acts and modeling negotiation subdialogues’, *Computational Linguistics*, **25**(1), 1–53, (1999).

[8] P. Cohen and H. Levesque, *Intentions in Communication*, chapter Rational Interaction as the basis for Communication, MIT Press, Cambridge, Massachusetts, USA, 1990.

[9] Florian Gallwitz, Maria Aretoulaki, Manuela Boros, Jürgen Haas, Stefan Harbeck, Richard Huber, Heinrich Niemann, and Elmar Nöth, ‘The erlangen spoken dialogue system EVAR: A state-of-the-art information retrieval system’, in *Proceedings of the 1998 International Symposium on Spoken Dialogue (ISSD'98)*, pp. 19–26, Sydney, Australia, (1998).

[10] Hans Kamp and Uwe Reyle, *From Discourse to Logic*, Kluwer, Dordrecht, 1993.

[11] Manfred Pinkal, ‘On semantic underspecification’, in *Computing Meaning*, eds., H. Bunt and R. Muskens, 33–55, Kluwer Academic Press, (1999).

[12] Massimo Poesio and David Traum, ‘Conversational actions and discourse situations’, *Computational Intelligence*, **13**(3), 309–347, (1997).