Combining the Practical Syllogism and Planning in Dialogue

Günther Görz, Alexander Huber, Bernd Ludwig, Peter Reiss
University of Erlangen-Nuremberg
Computer Science Institute

Abstract

We present a dialogue model which relates discourse relations and intentions by reasoning about pragmatic capabilities of dialogue particpants. For that purpose, planning approaches for discourse and application domain are employed. In this way, a computationally tractable version of the pratical syllogism is devised.

1 Rational Dialogues

Our goal is to build dialogue systems for rational interaction. Users can interact with the system in a given (ideally open) domain by conducting spoken dialogues. In principle, it should be possible to augment them by other forms of multi-modal interaction like gestures or the selection of items from a menu on a screen. Interactions are called "rational" because we want to apply rationality principles (at the knowledge representation level) to optimally select appropriate communicative actions. We assume that the satisfaction of user goals within the thematic framework of a particular application domain is to be achieved with the help of a dialogue system proper in cooperation with a technical application which we also call the "domain problem solver". Such a technical application can be an information or reservation system, a system for controlling certain devices, etc.

For dialogue modelling, we will follow a plan-based approach which has its roots in natural language processing and Artificial Intelligence. It provides the means to conduct task- or goal-oriented dialogues which are focussed on accomplishing concrete tasks as mentioned in the introduction. We claim that only a general planning approach enables cooperative response behaviour (pragmatic adequateness, overanswering) and the ability for negotiation. For the reasoning part, i.e. knowledge representation and inference for the interpretation of dialogue as well as for planning to satisfy user goals in the application domain, we insist on a clear commitment to a computational logic framework, in particular description logics. Of course, humans act incoherently and even inconsistently, and common sense reasoning can only to a certain extent be understood in terms of logic, but we are convinced that a coherent and consistent rational reconstruction is the best we can do about it. Such a constructive perspective has the advantage of enabling us to begin with a well understood framework for knowledge representation and reasoning upon which we can attempt to build rule systems for still idealized, but more realistic patterns of argumentation in specific domains. We believe that there is a potential to succeed in a variety of prevailingly instrumentalized contexts as it is the case with technical applications or in forensic argumentation.

Taking these claims serious, obviously a variety of complex issues must be addressed within the framework of our dialogue system, as, e.g., intention recognition, cooperativeness, grounding, or sharing plans – to name just a few important ones. For that, we refer to other publications of our group, e.g., (Görz et al., 2002; Bücher et al.,

2002; Ludwig et al., 2002). In this paper, we are adressing only one specific problem:

2 Choices in Rational Dialogues

In a series of papers, Asher and Lascarides introduced Segmented Discourse Representation Theory (SDRT) as an extension to Kamp's DRT and used it to model dialogue by combining compositional semantics, discourse structure and information about the participant's intentional states. The reason for their approach is to overcome inferential problems encountered in AI approaches to plan recognition in dialogue systems (Asher and Lascarides, 1997). A discourse is represented in the form of a SDRS (Segmented Discourse Representation Structure) which is a recursive structure of DRSs, i.e. semantic representations of linguistic expressions on the clause level, connected by rhetorical relations. Examples for such relations are explanation, contrast, or continuation. For discourse analysis, the authors propose to construct two different SDRSs, one for each discourse participant, in order to take their different cognitive states – we prefer to talk about epistemic states – into account. As a formal means for reasoning from the epistemic states of participants to what they say and vice versa, Asher and Lascarides introduce a version of Aristotle's Practical Syllogism: it "states that normally, people intend to do things that they believe help them achieve their goals"[1]. Under the rationality assumption for dialogues we propose for the multi-agent framework and the applications we work on within it, we will use a tighter, i.e. monotonic version of the Practical Syllogism by dropping "normally": If (a) participant B wants ψ and believes $\neg\psi$, and (b) B believes he can infer ψ from his knowledge base augmented by ϕ, and moreover ϕ is B's *choice* for achieving ψ, then (c) B intends ϕ.

There are two reasons for tightening up the Practical Syllogism into a monotonic version. The first is a technical, albeit quite important one: As already mentioned, in order to achieve a running system we committed ourselves to a particular computational logic framework, description logics, where the decidability – and furthermore an efficient implementation – of the inference problem is the foremost issue. Non-monotonicity in general would mean to lose decidability. The second reason is that in any specific application situation in fact the choice of an appropriate action is monotonic – either it is available or not. If not, a normality assumption is of little help. This decision can be modelled by a Reiter-style default rule, but it is a priori with respect to the application timepoint of the Practical Syllogism: Either counterevidence to the assumption that the chosen action is applicable exists in the actual situation, or it doesn't. So we have two distinct phases: First we have to check for counterevidence; if it exists, the *choice* cannot be executed and the Practical Syllogism is not applicable. If there is no counterevidence, we perform a monotonic derivation. Technically, the first step is represented in Abdallah's FIL calculus (Abdallah, 1995); it allows to express a certain situation description (model) with alternative model extensions where within each extended model we can infer monotonically. The monotonic version of Practical Syllogism now reads as follows:

$$\begin{array}{ll} \text{(a)} & (\mathcal{W}_B(\psi) \wedge \mathcal{B}_B(\neg\psi) \wedge \\ \text{(b)} & \mathcal{B}_B((\phi \rightarrow \psi) \wedge \mathit{choice}_B(\phi, \psi)) \\ \text{(c)} & \rightarrow \mathcal{I}_B(\phi) \end{array}$$

With the assumptions of Grice's maxims of cooperation and sincerity, the Practical Syllogism plays an essential part in the reasoning behind how responses to questions are interpreted in dialogue.

In Asher's and Lascarides' original version the choice operator is left open – there is just an existence assumption. To fill this gap, we propose a constructive approach: What we will argue for in the following is that we need to recur to planning in the application domain to provide a precise meaning for the *choice* operator in the rule above. The operationalization of the choice operator in terms of planning will exhibit the respective options in the actual search space, which applies to discourse as well as to domain operations, and furthermore provide an effective decision procedure. Dealing with both kinds of operations in a uniform way depends on how they are modelled in our system: in the underlying conceptual hierarchy both are represented formally in the same way.

[1] (Asher and Lascarides, 1997), p. 16 of the preprint version

Without any doubt, plans are an important source about discourse structure, as has been demonstrated by Grosz' and Sidner's studies as well as other AI work on dialogue (cf. also (Brietzmann and Görz, 1982)). We agree with Asher and Lascarides in their criticism that there is no direct way to structure dialogues satisfactorily by a global planning approach, because there isn't a one to one mapping between dialogue and plan structures. Our approach[2] as well as theirs is built upon the conviction that discourse interpretation requires intention recognition, and therefore a semantic-based theory of discourse structure is needed to assess when exploiting plans is appropriate. We have to distinguish between a domain level plan dealing with actions and a discourse level plan dealing with speech acts[3]. What we need is to predict automatically when the intentional structure of the discourse is isomorphic to the commonsense plan, and when it isn't; this requires a formal representation of semantics. Otherwise, because there is no strict isomorphism between both structures in general, wrong predictions about possible antecedents for anaphors in utterances that continue the dialogue may result.

The thesis proposed in this paper is that the key to a solution to this prediction problem lies in grounding the *choice* operator in the semantics of the resp. application domain. With respect to discourse structure, the role of action planning on the domain level is not to provide a global discourse segmentation, but rather to assign a precise meaning in terms of domain semantics to the *choice* operator in each situation where it is applied. The intended (perlocutionary) effect of a speech act constitutes a planning goal in the domain; it is determined further by the options to act in a particular cooperation context, e.g. by conjunctive subgoals. Whether there exists a precedence relation between subgoals in the conjunctive case can immediately be taken from the task plan in which possible dependencies are represented based on domain knowledge, resulting in a temporal order for the execution of subtasks. This matches well with Asher's and Lascarides' observation that a transition to the epistemic level, i.e. from structural relations between actions in commonsense plans to the level of knowledge, cannot provide a general solution, because the order in which facts are known by the dialogue participants usually doesn't matter. Only on the level of domain knowledge it can be decided whether dependencies between subgoals exist and in which order the corresponding tasks have to be executed.

In the following sections we will point out how this claim can be implemented in the context of an example domain that can be formalized with (classical) planning languages – in our case PDDL (Ghallab et al., 1998). PDDL is decidable and therefore computationally tractable, and there are several efficient planners implemented whose output – as discussed below – delivers the pragmatic options which are subject to the *choice* operator.

In our view, discourse segmentation is a consequence from planning actions in a setup where partners may exchange information about common goals. From a perspective complementing Sadek's work (Bretier and Sadek, 1996) on why partners have common goals at all, we elaborate an effective model of dialogues that explains the analysis and generation side of rational dialogues in human-computer interaction. In an application domain, there are several sources for a *choice* to be made by a dialogue participant:

- The derivation of a common goal from a natural language contribution to a dialogue.
- There may be more than one unique plan to achieve the common goal.
- It is possible as well that no plan can be found. In this case, *choice* is between relevant alternative contexts (cf. (Sperber and Wilson, 1995)) that can be computed effectively taking certain decision criteria into account (e.g. the usefulness of a alternative with respect to the initial intention of the speaker).
- During execution of a plan, the dialogue participant may run into trouble when assumptions that are vital for the plan to be carried out completely are violated.

Throughout the remainder of this paper, we try to illustrate our approach with the help of examples

[2] as described e.g. in (Görz et al., 2002; Bücher et al., 2002; Ludwig et al., 2002)

[3] (Asher and Lascarides, 1997) p. 5 of the preprint version

taken from a prototypical domain we have implemented: we use a model train to implement cooperative user interfaces to complex technical systems. In our scenario, a coffee machine is controlled by a PC as well as a robot that can load and unload cups from railway waggons and position it underneath the spout of the coffee machine that fills them with coffee. The cups are transported to destinations specified by the user.

3 Handling Underspecification in User Utterances

Our goal is to use a dialogue system for spoken language to control technical application systems. Let us consider the scenario described above. In very simple cases, of course there is no need for planning at all – the choice of the appropriate action is obvious. E.g., if only one switch exists, and this switch has only a set of predefinded states, the utterance "set the switch to position 1" (where 'position 1' is one of the defined states) has only one sensible corresponding system command.

But usually, the environment is more complex – there exists more than one switch and railtrack, several trains are moving with different speeds, etc. Now, the same command may intend different actions at the application level, depending on (a) the current state the application system is in, (b) possible user preferences, and (c) the fact that other requests may still be in the queue for processing. So, there is a 1:n-mapping between an utterance and possible commands to the technical application system, which is a kind of incomplete knowledge for the dialogue manager.

Because the user is free to let his descriptions for domain operations underspecified, utterances like "faster please" are possible. Here, not only the device is not specified (it could be one of several trains or even the robot), but also the degree of speed. The challenge for the dialogue system is to react in a sensible and cooperative manner. One way to resolve the underspecified items would be to initiate a clarification dialogue. But the system should always be as cooperative as possible, and before further inquiry, user preferences should be used as an additional source of knowledge, as well as the current state of the application environment. Taking these sources of knowledge into account, it is possible to determine a precise command for broad range of utterances[4].

We believe that planning can be used to determine the user's intentions and to choose the system action he probably wanted, taking into account the application state and his preferences.

Every plan has an initial state, a goal state and a set of possible actions. These actions are defined as plan operators, where each operator has its required initial state and one or more effects. When trying to find a plan, the initial states of the operators are matched with a representation of the current state of the application and of the current user with his (general) preferences. The intended application state is encoded in the user's utterance, but depends also on his preferences.

So, in the domain model, plan operators for different application states have to be defined. The set of (initial and goal) states is limited by the capabilities of the technical system that is addressed. The user preferences are encoded in the definitions of the plan operators, too. So it is clear that only underspecifications that are (directly or indirectly) handeled in the definitions of the plan operators can be filled. One simple example for encoding user preferences in a plan operator would be:

```
(:action faster
  :parameters  (?engine1 ?user)
  :precondition (not (maxspeed engine1))
  :effect (and
           (when (likesFast ?user)
             (maxspeed ?engine1))
           (when (likesSlow ?user)
             (mediumspeed ?engine1))))
```

Now, the command "faster please" leads to an application state where the train is running with maximum speed, if the user likes it fast. If the user's preference wrt. speed is "slow", the train will only move with medium speed. In this example, the application status is not regarded. But the operator definition can be expanded. For example, the current speed can be increased by two steps if the user likes it fast and by one step otherwise. The speed cannot be increased, if the train already runs at maximum speed.

[4] But there will still be cases where a set of actions of the technical system corresponds to one particular utterance. In those cases, it depends on the configuration of the dialogue system, wheter a clarification dialogue is initiated or e.g. a randomly choosed action is performed.

4 How is the Discourse Related to the Real World Situation?

The type of discourse situations we are considering here in a multi-agent framework is characterized by the following prominent factors: First, dialogue participants behave according to rational principles of conversation and action. Second, dialogues follow a certain rationale which in turn is determined by the intention to elaborate and execute joint plans (cf. (Chu-Carroll and Carberry, 1995; Chu-Carroll and Carberry, 1996; Carberry and Lambert, 1999)). As a consequence, dialogues are considered to be a means of exchanging information and requiring other dialogue participants to execute certain tasks defined in the application domain.

In the light of these guidelines of dialogue analysis (and generation), choices in a dialogue are determined by options while planning and executing plans for joint tasks. Furthermore, these options are constrained by the capabilities of a dialogue participant to act in his or her environment.

Options can appear on two levels in rationale dialogues: first, when reasoning about the effects of a speech act with respect to a model of interaction, and, second, when reasoning about the content of a speech act. Reasoning about content has many aspects: syntax, semantics and pragmatics not the only but the most prominent ones. The focus of this paper is on pragmatics and its effects on reasoning about the state of interaction.

(Carberry, 1990; Lambert, 1993; Lesh et al., 1999), among others, have pointed out that reasoning about plans is a key to understand dialogues; nevertheless, their work has – to the best knowledge of the authors – never been integrated with the work on planning problem solvers in AI in order to achieve an implementation of their discourse models that can reason efficiently and can be configured in a simple fashion to new applications. This paper tries to gap the brigde between linguistic theory and theoretical and practical work on planning by exploiting the expressivity of PDDL for the definition of application (and discourse) domains. In order to make planning applicable to dialogue processing, the following issues have to be addressed:

Planning must be bidirectional. The problem here is that PDDL operators must be applied for plan recognition as well as for plan generation. As we model collaboration in the sense of (Chu-Carroll and Carberry, 1996), application domain operators must be defined for each perspective in the modelled collaboration. In the applications we consider, normally user and system play complementary role: the user wants the system to perform some task and the systems tries to find and execute a plan that fulfils the task. As the task is to be defined in terms of a planning goal, natural language processing must construct it from utterances. To do that, we determine whether the semantic representation of the utterance talks about objects, states or actions. In the case of states and objects, a plan has to be computed if the information given in the utterance is not entailed in the current application situation; in the case of actions, we have to compute their possible effects by applying plan operators in a forward fashion. Of course, in order to avoid infinite recursion this process has to stop after a finite (small) number of iterations and therefore limits the system's capabilities to foresee the consequences implied by the user utterance. By planning in a forward direction, the system tries to get an imagination of what the user wants to happen. Ambiguities arise when the applied operators have conditional effects. In such a case, a decision procedure has to be applied that tries to get a good guess of what continuation would have the most positive and less risky effects on the user's intentions.

Maxims of conversation in the sense of (Grice, 1969) control planning. The need for a decision procedure is a consequence of the system to follow these general principles that of conversation. In our current implementation, there is no reasoning about these principles; therefore, the decision procedure always tries to fulfill user requests as fast as possible. In the worst case ambiguities cannot be resolved, and clarification is requested from the user. Otherwise, the completion of the task would have to be cancelled.

Planning in dialogues must be interactive. This is also due to the fact that planning is always interleaved with plan execution and there is no way to guarantee that each action in a plan can be ex-

ecuted as expected. Differences between planned and observed states motivate the need for replanning in order to fulfill obligations to satisfy user requests. In this way, handling conflicts can be reduced to the problem of controlling planning that was discussed above.

5 Examples for Computing *Choice*

The behaviour of our system, as it has been described in general terms up to now, is discussed in several examples showing when options come up and how they are handled.

In our example domain, for the sake of an intuitive example, we consider different reactions of the hearer to the speaker requesting *"Please bring me a cup of coffee!"*

In the first case, the setup is as follows: The robot – in station C – has one cup stored, an engine is positioned in station B, a waggon at a siding, whereas the coffee machine is in station C. Now, the speaker's request has to be translated into a planning goal. Obviously, the utterance is underspecified with respect to the destination. Here, two levels of *choice* have to be considered. On the content level, there are several options for destinations (all stations in our case); on the level of controlling the execution of a joint plan, there is a *choice* between deciding autonomously for an option, or to clarify this issue in interaction with the other dialogue participant:

- *You are in station A. I will bring the coffee to station A.*
- *Where do you want the coffee to be delivered? Is station A ok?*

As one can observe immediately, there are numerous options for speech acts and text generation when the need for clarification is verbalized.

```
(:init (at engine1 stationB)
       (at waggon1 siding)
       (cup-state cup1 empty)
       (stored-on cup1 stack1)
       (coffee-machine-state cm1 off)
       (at cm1 stationC)
       (at cup1 stationC))
(:goal (at cup1 stationA))
```

The presentation of the computed options are intended to show that the major issue to be addressed is underspecification, not non-monotonicity. As far as discourse planning is concerned, our approach is to find (disjunctive) alternatives on the basis of observed facts instead of finding contradictions to default assumptions. This observation leads to our claim that a monotonic practical syllogism is sufficient.

In the example, to resolve the underspecification, a decision is made in favour of station A as the destination. As the translation of the request shows, for the interpretation of an utterance contextual information is taken into account as well as the content of the utterance. We have configured the IPP Planner (Koehler et al., 1997) with plan operators for the functionality of the model train domain. The domain plans shown in this paper are always computed by the IPP Planner. IPP finds the following solution for the request:

```
0: switch-on cm1
   put-below-spout cup1 cm1 stack1
   compute-route stationB siding engine1
1: go engine1 stationB siding
   fill-cup cm1 cup1
2: connect waggon1 engine1 siding
   compute-route siding stationC engine1
3: go engine1 siding stationC
4: put-on-waggon engine1 waggon1 cup1
                 CM1 stationC
   compute-route stationC stationA
                 engine1
5: go engine1 stationC stationA
```

In the next setup, there are two cups: somebody put a cup (*cup2*) underneath the spout and prevents the plan from being executed[5] These constraints are added to the initial situation:

```
(:init (at cm1 stationC)
       (at cup1 stationC)
       (underneath-spout cup2 cm1)
       (at cup2 stationC)
       (cup-state cup2 full))
```

Observe that additional constraints in the application domain influence the course of the dialogue. Starting in the same way as above, now a conflict of the plan with external events has to be handled. Again, *choice* is between options on the application as well as on the control level:

[5] In general, the case of implicit conflicts must be handled: A user may give a precise command that could be satisfied directly by the system. But its execution may conflict with a former wish or preference (e.g. delivering the coffee with train 1 may block the transport way connection for train 2). So, an execution of a wish may or may not have side effects that may be desired, irrelevant, or should be avoided.

- *There is a cup underneath the spout. Your request is cancelled.*
- *There is a cup underneath the spout. It's currently being filled. Do you want this cup?*
- *There is a cup underneath the spout. It will be removed immediately. Is it ok for you to wait a few seconds or do you prefer to cancel your request?*

How are conflicts related to the practical syllogism? Again, we claim that they give no argument for non-monotonicity. In the sense of Abdallah's partial logic (see (Abdallah, 1995)), in a logical reconstruction of what was expected to hold this situation and what was observed, of course a logical contradiction is entailed. But only up to the point when – as it comes out now – the wrong option was choosen, or – in Abdallah's words – our "justification" knowledge was not correct. This means, we just have to leave the wrong path from history to future and follow the right one to remain in a "monotonic world".

In the last setup, we have to handle the following situation: There are two cups in station C, one of them underneath the spout and filled, the other one on the robot's stack. In contrast to the last example however, the user's request for a cup of coffee was interpreted to be underspecified as far as the selection of a cup is concerned. As a consequence, the goal contains a disjunction of all the known cups as possible candidates for satisfying the user request.

```
(:goal (or (at cup1 stationA)
           (at cup2 stationA)))
```

Now, with this underspecification in the goal, a plan is found that may satisfy the user's request. The output of the planner below indicates in step 6 that there are other options that could be an answer to the request as well.

```
0: compute-route stationB siding engine1
1: go engine1 stationB siding
2: connect waggon1 engine1 siding
   compute-route siding stationC engine1
3: go engine1 siding stationC
4: put-on-waggon engine1 waggon1 cup2 cm1
                 stationC
   compute-route stationC stationA
                 engine1
5: go engine1 stationC stationA
6: GOAL-REACHED
```

So, correct and useful utterances in the spirit of (Webber, 1986) include:

- *I will bring you cup2. It's filled already.*
- *Is cup2 ok for you? Or do you prefer cup1? It would take me a bit longer, however.*
- *Do you want cup1 oder cup2?*

In each of the example cases discussed above, a particular discourse relation is assigned to each option available. As a consequence, the user's reaction determines how the structure of the dialogue will develop in further steps to come: Depending on the user's *choice* for a proposed option, the current discourse and application situation have to be updated and modified differently. New constraints that result from this update process, may in turn influence the *choices* available on discourse and application level.

6 Conclusions

We argue that the *choice* operator leads to a dialogue model that distinguishes between discourse and application domain as choices may become neccessary between options in the application (how to satisfy a request?) and in the discourse (how to communicate options?). Another consequence is that dialogues are guided by two levels of control for analysis as well as for generation: first, the computation and selection of options in the discourse (for explaining at least semantic, syntactic and discourse pragmatic ambiguities) and second the computation and selection of options in the application depending on decisions made in the discourse domain. Effective selection of options implies the availability of a decision procedure for this task (see (Carletta, 1992)). It must be able to handle ambiguities as well as unsatisfiable intentions. In our opinion, a (computationally tractable) approach requires different algorithms for reasoning and deciding due to the different nature of the tasks to be solved.

In our example application, we try to integrate reasoning and decision procedures in a complete dialogue system for spoken langugae processing. On the basis of the approach outlined in this paper, the knowledge for making the control levels work,

can be set up for various applications by defining the functionality of the application domain in terms of a set of PDDL plan operators.

There is a lot of related work on the implementation of dialogue systems. The most important one seems to be the TRAINS system and its successors as described in (Allen et al., 2001). In our view, a distinguishing feature of our work is the focus on data-driven adaptability to new domains – an issue not discussed in very much detail in (Allen et al., 2001). (Yates et al., 2003) describe a system that responds to user requests for handling a VCR. As our system, it computes plans to satisfying user intentions. However, the paper does not explain if options – as discussed above – can be computed and how they are integrated in a dialogue. We find that our hybrid approach covers a broader range of dialogues.

References

A. N. Abdallah. 1995. *The Logic of Partial Information*. New York.

J. F. Allen et al. 2001. Towards Conversational Human-Computer Interaction. *AI Magazine*, 22(4): 27–38.

N. Asher, A. Lascarides. 1998. Questions in Dialogue. *Linguistics and Philosophy*, 21(4): 237–309. Preprint version at: http://www.utexas.edu/cola/depts/philosophy/faculty/asher/main.html.

P. Bretier, D. Sadek. 1996. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. In: Müller, J. P. et al. (ed.): *Intelligent Agents III – Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, Lecture Notes in Artificial Intelligence: 189–203. Heidelberg.

A. Brietzmann, G. Görz. 1982. Pragmatics in Speech Understanding – Revisited. In: J. Horecky (ed.): *COLING 82 – Proceedings of the Ninth International Conference on Computational Linguistics*: 49–54. Amsterdam.

K. Bücher, G. Görz, B. Ludwig. 2002. Corega Tabs: Incremental Semantic Composition. In: Görz, G. et al. (ed.): *KI-2002 Workshop on Applications of Description Logics CEUR Proceedings*. Aachen.

S. Carberry. 1990. *Plan Recognition in Natural Language Dialogue*. MIT Press.

S. Carberry, L. Lambert. 1999. A process model for recognizing communicative acts and modeling negotiation subdialogues. *Computational Linguistics*, 25(1): 1–53.

J. Carletta. 1992. *Risk-Taking and Recovery in Task-Oriented Dialogue*. University of Edinburgh.

J. Chu-Carroll, S. Carberry. 1995. Generating information-sharing subdialogues in expert-user consultation. In: *Proceedings of IJCAI 1995*: 1243–1250.

J. Chu-Carroll, S. Carberry. 1996. Conflict detection and resolution in collaborative planning. In: *Intelligent Agents: Agent Theories, Architectures, and Languages*: 111–126. Berlin.

M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins. 1998. *PDDL – The Planning Domain Definition Language*. AIPS-98 Planning Committee.

G. Görz, K. Bücher, Y. Forkl, M. Klarner, B. Ludwig. 2002. Speech Dialogue Systems – A "Pragmatics-First" Approach to Rational Interaction. In: Menzel, W. (ed.): *Natural Language Processing between Linguistic Inquiry and System Engineering*. Festschrift für Walther von Hahn (in print). Preprint version at: http://www8.informatik.uni-erlangen.de/IMMD8/staff/Goerz/papersgg/vHahnColl_2002.ps.gz. Hamburg.

H. P. Grice. 1969. Utterer's Meaning and Intention. In: *Philosophical Review, Vol. 78*: 147–177.

J. Koehler, B. Nebel, J. Hoffmann, Y. Dimopoulos. 1997. Extending Planning Graphs to an ADL Subset. In: *Proceedings ECP-97*: 273–285. Berlin.

L. Lambert. 1993. *Recognizing Complex Discourse Acts: A Tripartite Plan-Based Model of Dialogue*. University of Delaware.

N. Lesh, C. Rich and C. L. Sidner. 1999. Using Plan Recognition in Human-Computer Collaboration. In: *Proceedings of the 7th International Conference on User Modelling*: 23–32. Banff.

B. Ludwig, K. Bücher, G. Görz. 2002. Corega Tabs: Mapping Semantics onto Pragmatics. In: Görz, G. et al. (ed.): *KI-2002 Workshop on Applications of Description Logics CEUR Proceedings*. Aachen.

D. Sperber, D. Wilson. 1995. *Relevance – Communication and Cognition*. Oxford.

B. L. Webber. 1986. Questions, answers and responses: Interacting with knowledge-base systems. In: Brodie, M. (ed.): *On Knowledge Base Management Systems*: 366–402. New York.

A. Yates, O. Etzioni, D. Weld. 2003. A Reliable Natural Language Interface to Household Appliances. *Proceedings International Conference on Intelligent User Interfaces*: 189–196. Miami, Florida.