Tracing Actions Helps in Understanding Interactions

Bernd Ludwig
Chair for Artificial Intelligence, University of Erlangen-Nürnberg
Am Weichselgarten 9, D-91058 Erlangen
Bernd.Ludwig@informatik.uni-erlangen.de

Abstract

Integration of new utterances into context is a central task in any model for rational (human-machine) dialogues in natural language. In this paper, a pragmatics-first approach to specifying the meaning of utterances in terms of plans is presented. A rational dialogue is driven by the reaction of dialogue participants on how they find their expectations on changes in the environment satisfied by their observations of the outcome of performed actions. We present a computational model for this view on dialogues and illustrate it with examples from a real-world application.

1 A Pragmatics-First View on Dialogues

Rational dialogues that are based on GRICE's maxims of conversation serve for jointly executing a task in the domain of discourse (called *the application domain*) by following a plan that could solve the task assigned to the participants of the dialogue. Therefore, the interpretation of new contributions and their integration into a dialogue is controlled by global factors (e.g. the assumption that all dialogue participants behave in a cooperative manner and work effectively towards the completion of a joint task) as well as by local factors (e.g. how does the new contribution serve in completing the current shared plan?).

Ony if these factors are represented in an effective and efficient formal language, dialogue systems can be implemented. Examples of such models and their implementation are the information-state-update approach (an implemented system is described in (Larsson, 2002)), or – more linguistically oriented – approaches like the adjacency-pair models or intentional models such as GROSZ and SIDNER's (see (Grosz and Sidner, 1986)).

Even if it has been noted often that discourse structure and task structure are not isomorphic, only a few contributions to dialogue research focus on the question of how both structures interfere (see Sect. 2). In this paper, we emphasize that it is important to distinguish between the *dialogue situation* and the *application situation*: The former is modified whenever speech acts are performed, whereas the latter changes according to the effects of each action being executed. In this section, we will use a MAPTASK dialogue to show what the notions *dialogue situation* and *application situation* intend to mean. After presenting related work in Sect. 2, we present our approach first informally and then formally by explaining which AI algorithms we apply in order to turn the informal model into a computationally tractable one.

1.1 Talking about Domain Situations

The main hypothesis of this paper is that modifications of the dialogue situation are triggered by changes of the application situation. As a response to a speech act, dialogue participants perform a series of actions aiming at achieving some goal. If these actions can be executed, the reaction can signal success. At this point, our understanding of the role of shared plans exceeds that of (Grosz et al., 1999): GROSZ and KRAUS define an action to be *resolved* if it is assumed that an agent is able to execute the action. However, in order to understand coherence relations in complex dialogues, it is important to know whether an action has actually been executed and what effect it has produced. Consider the following excerpt from a MAPTASK dialogue (*MAP 9*, quoted from (Carletta, 1992)):

R: ++ and ++ you are not quite horizontal you are taking a slight curve up towards um the swamp ++ not obviously going into it

G: well sorry I have not got a swamp

R: you have not got a swamp?

G: no

R: OK

G: start again from the palm beach

G has failed to find the swamp, which means *G* has failed to perform the action necessary to perform the next one (take a slight curve).

In order to solve the current task, *R* has been able to organize a solution for the task at hand which may or may not involve the other dialogue participant *G*. How can *R* put his solution into action? First, he executes each step and, second, validates after each step whether all expectations related to it are fulfilled.

1.2 Talking about Error and Failure

In the example above, *R*'s expectations are not met because *G* does not find the swamp on the map. However, this would be a precondition for *R* to continue putting the solution into action that he has organized. On the other hand, *G* understands that finding the swamp is very important in the current task, but he missed to reach that goal. In order to share this information with *R*, *G* verbalizes his failure diagnosis: *"I have not got a swamp."*

This turn makes *R* realize that his solution does not work. Obviously, *R* believed his solution to be well elaborated because he tries to get a confirmation of its failure by asking back *"you have not got a swamp?"* *G*'s reacknowledgement is a clear indication for *R* that it is necessary to reorganize his solution for the current task. Being a collaborative dialogue participant, he will try to recover from that failure to explain the way to the destination.

1.3 Domain and Discourse Strategies

For the purpose of recovery, the dialogue participants try to apply a repair strategy that helps them to reorganize the solution. Repair strategies are complex domain dependent processes of modifying tasks and solutions to them. Even being domain dependent in detail, there are some strategies that are domain independent and are regularly adapted to particular domains:

- **Delay**: Maybe it is the best decision to wait a bit and try the failed step again.
- **Delegation**: Maybe someone else can perform better.
- **Replanning**: Another solution should be found based on the current error diagnosis.
- **Relaxation**: Modify some parameters or constraints of the task so that a tractable solution can be found.
- **New Tools**: Maybe somehow the dialogue participant can extend his capabilities in the domain so that he can achieve the solution using other, more, or stronger tools and means.
- **Negotiation**: Try to retrieve new helpful information from the user or to come to an agreement of how the task can be modified.
- **Cancellation**: Sometimes giving up to find a solution is the only remaining possibility.

This list is necessarily incomplete as depending on the particular domain and current situation in which a dialogue participant has to act these strategies appear in very different fashion. So, it is hard to decide whether exception handling for a single case is taking place or if a particular strategy is being applied. In the example dialogue, *G* tries to suggest a `replanning` by telling to *R* up to what point he was able to understand *R*'s explanations.

According to his communication strategy, a dialogue participant tells his deliberations in more or less detail, sometimes even not at all. This is the case in the example dialogue above. In the last turn, *G* does not tell that he wants *R* to reorganize his solution. *R* must infer this from the content, in particular from the request to restart the explanation at a point that has been passed before the *G* had failed to understand a step in *R*'s explanation.

This example shows that domain strategies and communication strategies interfere in a dialogue and that complicated reasoning is necessary to identify them in order to react appropriately.

Our analysis shows that the notion of coherence is strongly related with the execution of single steps in a solution. Often, coherence cannot be explained satisfactorily within a discourse, but the current situation in which an utterance is made, must be taken into consideration as well.

2 Related Work

There are several main research directions on dialogue understanding. The one closest to our approach is activity-based dialogue analysis (Allwood, 1997; Allwood, 2000) contrasting BDI-style approaches such as the one by (Cohen and Levesque, 1995). This research shows how speech acts are related to expectations expressed by means of language and inspired our approach. However, ALLWOOD does not work out in detail how the pragmatics of the application domain can be formalized in a tractable way. (Carletta, 1992) shows in a corpus analysis that risk taking is a elementary behavior of dialogue participants. (Bos and Oka, 2002) uses first-order logic in a DRT environment to reason about the logical satisfiability of a new utterance given a previous discourse. For reasoning about action however, we think that a first-order theorem prover or model builder is not the ideal tool because it is too general. Additionally, in dialogues about acting in an environment, the primary interest of semantic evaluation is not whether a formula is true or false, but how a goal or task can be solved. Therefore, planning is more appropriate than proofing formulae. Work on planning as part of dialogue understanding is reported in (Zinn, 2004). This paper does not address selecting strategies for error recovery. Conflict resolution is addressed in (Chu-Carroll and Carberry, 1996). However, the presented discourse model is not computationally effective. (Huber and Ludwig, 2002; Ludwig, 2004) present an interactive system which uses planning, (Yates et al., 2003) and recently (Lieberman and Espinosa, 2006) reported on applying planning as a vehicle for natural language interfaces, but none of the papers discusses how a dialogue can be continued when a failure in the application occurs. In the WITAS system (see (Lemon et al., 2002)), *activities* are modelled by *activity models*, one for each type of activity the system can perform or analyse. A similiar recipe-based approach is implemented in COLLAGEN (Garland et al., 2003). As activities are hard-coded in the respective model, adaptation of the task and dialogue structure to the needs in a current situation are harder to achieve than in our approach in which only goals are specified and activities are selected by a planner depending on the current state. In addition, executing plans by verifying preconditions and effects of an activity that has been carried out recently lies the basis for a framework of understanding the pragmatics of a dialogue that is not implemented for a particular application, but tries to be as generic as possible.

3 Problem and Discourse Organization

A computational approach that aims at analyzing and generating rational – i.e. goal-oriented – dialogues in a given domain must address the issues of organizing a solution in the application domain as well as in the discourse domain. Furthermore, it must provide an effective method to organize solutions, classify current states in the discourse as well as in the application situation (are they erroneous or not?) and select strategies that promise a recovery in case of an error.

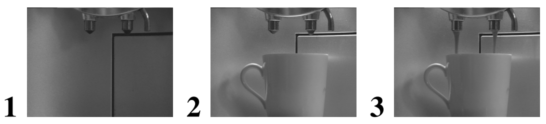

Figure 1: Example data for a classification task.

3.1 Expectations and Observations

To diagnose an error, a dialogue participant must be able to determine whether his expectations on how the environment changes due to an action match his observations.

3.2 The Origin of Expectations

The expectations of a dialogue participant are derived from his organization of a solution to the current task. Each step herein has – after it has been executed – a certain intended impact. It forms the expectations that are assigned to a single step.

An expectation is met by a set of observations if the observations are sufficient to infer the expectation from. The inference process that is employed in this context may be as simple as a slot-filling mechanism or as complicated as inference in a formal logic. In the slot-filling case, the inference algorithm is to determine whether the semantic type of the answer given by the user match the type that was expected by the dialogue system.

However, inference in the sense of this paper may involve difficult computations: Expectations are generated while a solution is organized. Each step in a solution leads to certain changes in the environment that are expected to happen when the step is actually executed. Later in the paper, we will demonstrate how planning algorithms can generate such expectations. Additionally: – see Fig. 1) – in order to verify expectations of the request *"Fill coffee into the cup!"* image data need to be classified before it can be concluded that the expectation (image 3) is satisfied.

4 Planning Solutions

In order to illustrate our approach how a natural language dialogue system can organize solutions for user requests, we discuss a natural language interface for operating a transportation system. The

```
produce-coffee
:parameters (?c - cup ?j - jura)
:precondition
  (and (under-spout ?c)
       (not (service-request ?j)))
:effect (and (not (empty ?c)) (ready ?j))
```

Figure 2: Example of a plan operator in PDDL

system allows to control a model train installation and electronic devices currently on the market.

4.1 Organizing a Solution

First of all, in order to specify the (pragmatic) capabilities of the whole system, a formal model of the system is needed that allows the necessary computations for organizing solutions. For this purpose, we model all functions provided by the system in terms of plan operators in the PDDL planning language. Fig. 2 shows an example.

This operator describes part of the functionality of the automatic coffee machine that is integrated into our system: the function `produce-coffee` can be executed if there is a cup under the spout of the machine and if it does not require service (as such filling in water or beans). These are the preconditions of the function. After coffee has been produced, it is expected that the environment is changed in the following way: the cup is not empty any longer, and the machine is ready again.

In order to organize a solution, a task is needed and knowledge about the current state of the environment. The latter comes from interpreting sensor data, while the former is computed from natural language user input. For the example request *“Fill in a cup of espresso!”*, we assume the current state in Fig. 3 to hold and use the formula in Fig. 4 as the description of the current task to solve.

The example in Fig. 3 assumes that the cup is parked and empty, and the coffee machine and the robot (used for moving cups) are ready. The task is formalized as a future state of the environment in which the cup is parked and the coffee machine is in the mode *one small cup* (see Fig. 4).

To compute a solution, a planning algorithm (we incorporated the FF planner (Hoffmann and Nebel, 2001) in our system) uses the information about the current state and the intended future state as input and computes a plan for a number of steps to execute in order to solve the task (see Fig. 5).

In the following, we will consider such a plan as in Fig. 5 as an organized solution for the task to be solved. Expected changes of the environment are defined by the effects of each step of the solution. Fig. 6 shows which changes are expected if the plan in Fig. 5 is eventually executed.

4.2 Executing a Solution

Given a plan for a task to be solved, our dialogue system executes each step sequentially. Before a step of the solution is performed, the system verifies each precondition necessary for the step to be executable. If all tests succeed, actuators are commanded to perform everything related to the current step. Feedback is obtained by interpreting sensor input which is used to control whether the intended effects have been achieved. For the function `produce-coffee` above, the following procedure is executed:

```
produce-coffee (cup c, jura j) {
  if test(under-spout,c)=false
    signal_error;
  else {
    if test(service-request,j)=true
      signal_error;
    else do produce-coffee, c, j;
  };
  if test(empty,c)=true signal_error;
  else {
    if test(ready,j)=false signal_error;
    else return;
  };
}
```

In this procedure, each precondition of the function `produce-coffee` is verified. If the system can infer from the sensor values that a precondition cannot be satisfied, it signals an error. The same is done with all effects when the actuators have finished to change the environment. As we will discuss in Sect. 6, these error signals are the basic information for continuing a dialogue when unexpected changes have been observed.

5 Diagnosing Errors

How can the dialogue system react if a precondition or effect does not match the system’s expectations? The primary goal of a dialogue system is to

```
(and (parked cup) (empty cup)
     (ready jura) (ready robo))
```

Figure 3: The current state of the environment for the example in Sect. 4

```
(and (parked cup) (mode-osc jura))
```

Figure 4: The task to be solved

```
put-cup-on-spout(cup,jura,robo)
draw-off-osc(cup,jura)
produce-coffee(cup,jura)
go-in-place(train)
take-cup-off-spout(cup,jura,robo)
load-cup-on-waggon(cup,jura,robo,train)
park-cup(cup,jura,robo,train)
```

Figure 5: A plan for the task in Fig. 4

Step #	Action and expected changes
1	**put-cup-on-spout(cup, jura, robo)**
	`(under-spout ?c)`
	`(not (robo-loaded ?r ?c))`
	`(not (parked ?c))`
2	**draw-off-osc(cup, jura)**
	`(not (ready ?j)) (mode-osc ?j)`
3	**produce-coffee(cup, jura)**
	`(not (empty ?c)) (ready ?j)`
4	**go-in-place(train)**
	`(in-place ?t)`
5	**take-cup-off-spout(cup, jura, robo)**
	`(not (under-spout ?c))`
	`(robo-loaded ?r ?c)`
6	**load-cup-on-waggon(cup, jura, robo, train)**
	`(not (robo-loaded ?r ?c))`
	`(train-loaded ?t ?c)`
7	**park-cup(cup, jura, robo, train)**
	`(not (train-loaded ?t ?c))`
	`(parked ?c)`

Figure 6: Expected changes in the environment

meet principles of conversation such as GRICE's maxims. Often, however, it is not obvious to the user how a particular constraint in a plan is related to the current task. Therefore, a plausible and transparent explanation of an error brings the diagnosed mismatch in its context of the current action and solution for the current task. At the core of each explanation are the unexpected observations. The context of the error is formed by all available sensor values and the history of past actions which are steps in the solution (see Fig. 5) for the current task. The example in Fig. 7 shows the state of the system after the first four steps of the solution introduced in Sect. 4.1 have been carried out. After executing `take-cup-off-spout` however, the observed changes of the system state do not match the expected ones: Analyzing the most recent images and the robot's weight sensor, the system recognizes that the cup still is on the spout, but not being carried away by the robot.

Given the situation in Fig. 7, what is the system able to or should communicate to the user? Without any additional reasoning, the system could only tell what it has observed:

> **User**: *Fill in a cup of espresso!*
>
> **System**: *The cup is under the spout and the robot is not loaded with the cup.*

Such a response would not be very helpful for the user (especially if he cannot see the situation or is unexperienced in handling the system). Our approach to generating plausible explanations for failures of the system is to compute a diagnosis that takes the context of the failure and the task to be solved into account. The approach relies on an explicit error model for each function that the system provides in the sense of (Balduccini and Gelfond, 2003): In the case of `take-cup-off-spout` the following symptoms and causes lead to failure of the function:

Symptom	**Cause**
`(under-spout c)`	wrong position
	robo lost cup
`¬ (robo-loaded r c)`	arm motor off
	arm not ok
	didn't hold cup

The approach of (Balduccini and Gelfond, 2003) generates models in a formal logic of what could be reasons for the failure of the system and in which state the system would be in each case. If more than one model is generated when an unexpected observation has been made, the system has to find evidence for each model and compare how reliable a diagnosis is (given the available sensor values and their interpretation). For that purpose, decision trees are trained on past cases of failure and the confidence values for the computed interpretations. (e.g. to which class does the image in Fig. 1?). Applying such a decision tree on a given system state and mismatched expectation results in a unique model that is seen as an explanation for the failure of the system's function. In this way, the dialogue system can explain better to the user what it thinks has happened during the execution of `take-cup-off-spout`:

> **User**: *Fill in a cup of espresso!*
>
> **System**: *I could not carry the cup to the train because I lost the cup on the way.*

From the viewpoint of discourse analysis, we can consider the mismatch between observations and expectations as the main cause for the system response. As a consequence, we can consider the logical relation between the current task, the solution organized by the system, and the computed error diagnosis as sufficient conditions for any discourse relation to hold between the user and the

Actions up to now	**System state** **after** `go-in-place`	**Expected state**	**Error Diagnosis**
`put-cup-on-spout`	`(ready jura)`		none
`draw-off-osc`	`(ready robo)`		none
`produce-coffee`	`¬(parked cup)`		none
`go-in-place`	`¬(empty cup)`		none
	`¬(robo-loaded robo)`		none
	`(mode-osc jura)`		none
	`(in-place train)`		none
	`(under-spout cup)`		none
Last Action	**Observed state**		
`take-cup-off-spout`	`(under-spout cup)`	`¬(under-spout cup)`	robo could not
	`¬(robo-loaded robo)`	`(robo-loaded robo)`	hold the cup

Figure 7: Context information for the diagnosis of an error

system utterance in the dialogue excerpt above: In terms of TRAUM's DU acts (Traum, 1994), coherence between both utterances is established as a `reject` relation as the purpose of the utterance is to indicate failure of the task that has been initiated by the user request. To explain the MAPTASK dialogue cited in the introduction, another level of pragmatic reasoning is required: As already mentioned in Sect. 1.3, the dialogue system is cooperative and tries to find out a way in order to nevertheless solve the task as completely as possible.

6 Error Repair and Discourse Update

Such a way out consists in applying a strategy that is appropriate for the current state of the system and the interaction with the user. In the AI (Mitchell, 1997) and robotics (Bekey, 2005) literature, algorithms for applying adaptive strategies in different situations are all based on the current state as input and an evaluation function that helps selecting an optimal strategy.

6.1 Repair Strategies in the Application

A favorite algorithm for this kind of interactive control problems is to select the optimal policy out of a set of possibilities. Before that, an evaluation function is trained by reinforcement learning to always select the action that maximizes the reward obtainable in the current state. In (Henderson et al., 2005), this machine learning approach was applied to selecting speech acts after training an evaluation function on a dialogue corpus in which each utterance was labeled with a speech act.

Different from (Henderson et al., 2005), in our approach the actions between whom the dialogue system can choose are repair strategies instead of speech acts. In our opinion, speech acts are a phenomenon of another invisible process – text generation – but not objects of the decision at the discourse planning level: the selection of a repair strategy does not fix the type of a speech act nor its content. The way a repair strategy works and – as a consequence – has influence on the flow of a dialogue is that, firstly, it modifies the current task and, secondly, seeks a new solution that will be executed later on. Future speech acts then are a result of performing single steps of the new solution.

To recover the `take-cup-off-spout` function, the system may have the option to fill another cup and try to bring this one to its destination. It must be noted, however, that this option depends to a large extent on the availability of another empty cup, the readiness of the robot and the coffee machine and sufficient resources like beans, water, and time to complete the task. All these parameters influence the computation of the reward and the risk to be assigned to this domain-specific variant of a *New Tools*-strategy (see Sect. 1.3).

6.2 Effects on Discourse Update

The MAPTASK dialogue in Sect. 1.1 even is somewhat more complicated: *G* understands that he does not have the capability to repair the misunderstanding as there is too much information missing. Therefore, he initiates a *Negotiation*-strategy in which he switches the topic of the dialogue to the domain of strategies for MAPTASK. *G* proposes a new strategy with a slightly modified task to *R*. It is exactly this logical relation that explains the coherence between the turns in this dialogue. In this case, the coherence cannot be established by reasoning in one single domain.

In terms of the Conversation Acts Theory by (Traum and Hinkelman, 1992) and (Poesio and Traum, 1998; Traum, 1994), the discourse segment related to the solution for a task can be called *multiple discourse unit* (MDU). Consequently, the conversation acts for MDU are a trace of the dialogue participant's decisions on which interactions are needed to solve the task and how they could be verbalized best. Argumentation is based on the formal knowledge about the domain, the current task, and a solution proposed for it. This means that an analysis of the current state of the system and the dialog provides facts that can be used as conditions for the applicability of a speech act. Equally, facts about the system are conditions for the applicability of a system function at a certain point of time. It follows directly from this observation that planning argumentation acts can be viewed as a special kind of classical planning in AI. However, due to the interactive nature of such a dialogue task, it must be possible to react flexibly and directly on mismatches between expectations and observations for speech acts and the intended changes during the course of a dialogue.

Therefore, in this paper dialogue management is seen as a special case of reactive planning. As shown above, discourse relations are derived from meta-information about the state of executing a plan for the current task. The discourse relations serve as preconditions for speech acts effectuating the update of the dialogue state.

6.3 Diagnosing Linguistic Errors

Our model of relating pragmatics and interaction can be extended to discourse pragmatics as well. It is particularly helpful to understand *grounding* acts in the `utterance unit` level (see (Traum and Hinkelman, 1992)). In this case, the ("application") domain is that of understanding language. The task to be solved is to extract words from a speech signal and to construct meaning from those words. Error diagnoses occur frequently and options caused by ambiguities of natural language have to be tested whether they can help to repair a diagnosed error automatically. If not, the diagnoses as symptoms of misunderstanding have to be assigned to possible causes. Strategic decisions have to be made how to communicate the causes and possible suggestion for repairs to the user. This reasoning results in *grounding* acts that would be hard to analyze otherwise. This idea can be applied to negotiating speech acts as well. The difficult task, however, is to implement a diagnosis algorithm for failure in syntax analysis, (compositional) semantics, and speech act analysis.

7 Understanding User Utterances

There are implications of our approach for computational semantics: In order to see whether a user utterance meets the system's expectations, it is necessary to analyze which domain the utterance refers to. For this purpose, expectations for discourse and system state are maintained separately. Each new contribution must satisfy the discourse expectations (e.g. an answer should follow a question) and pragmatic expectations (the content of the contribution must extend without contradictions what is known about the current solution. To test this, a model (in the sense of formal logic) is computed for the conjunction of the new content and the currently available information.

As discussed above, it may happen in dialogues that the focus is switched to another topic, i.e. another domain, and the coherence can be established only when taking this domain shift into account. In order to be able to detect such a domain shift, we define the meaning of performative words depending on whether they refer to the hidden reasoning processes that are part of our approach, the discourse control domain, or to states, objects, and functions in the current applications situation: In the MAPTASK example, the utterance *Start from the palm beach* refers to the process of strategy selection and organization of a solution, but not to the domain of explanations in a map.

8 Conclusions

The presented approach allows dialogue understanding to take into account that the (human) dialogue participant the system is interacting with is (at least) equally able to diagnose errors and mismatches between observations and expectations and generates utterances intended to update the dialogue state according to these findings. Therefore, for establishing the coherence of a user utterance, there are always several options: firstly, the user continues the current solution, secondly, he diagnoses failure and reports about it, and thirdly, he switches the focus to another domain including discourse update and repair strategies.

For these options, our approach devises a computational model able to explain dialogues in which coherence of turns is difficult analyze. In this way, more natural dialogues can be analyzed and generated. As the approach incorporates a model for how talking about actions is related to acting in a formalized domain, it serves as a basis for constructing natural language assistance systems, e.g. for a great range of electronic devices.

References

Jens Allwood. 1997. Notes on dialog and cooperation. In Kristina Jokinen, David Sadek, and David Traum, editors, *Proceedings of the IJCAI 97 Workshop on Collaboration, Cooperation, and Conflict in Dialogue Systems*, Nagoya, August.

Jens Allwood. 2000. An activity based approach to pragmatics. In Harry C. Bunt and B. Black, editors, *Abduction, Belief, and Context in Dialogue*, Studies in Computational Pragmatics. John Benjamins, Amsterdam.

Marcello Balduccini and Michael Gelfond. 2003. Diagnostic reasoning with a-prolog. *Theory and Practice of Logic Programming*, 3(4–5):425–461, July.

George A. Bekey. 2005. *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press.

Johan Bos and Tetsushi Oka. 2002. An inference-based approach to dialogue system design. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 113–119.

Jean Carletta. 1992. *Risk-Taking and Recovery in Task-Oriented Dialogue*. Ph.D. thesis, University of Edinburgh.

Jennifer Chu-Carroll and Sandra Carberry. 1996. Conflict detection and resolution in collaborative planning. In *Intelligent Agents: Agent Theories, Architectures, and Languages*, volume 2 of *Lecture Notes in Artificial Intelligence*, pages 111–126. Springer Verlag.

Phil R. Cohen and Hector J. Levesque. 1995. Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 65–72, San Francisco, CA, June.

Andrew Garland, Neal Lesh, and Charles Rich. 2003. Responding to and recovering from mistakes during collaboration. In Gheorghe Tecuci, David W. Aha, Mihai Boicu, Michael T. Cox, George Ferguson, and Austin Tate, editors, *Proceedings of the IJCAI Workshop on Mixed-Initiative Intelligent Systems*, pages 59–64, August.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Barbara J. Grosz, Luke Hunsberger, and Sarit Kraus. 1999. Planning and acting together. *AI Magazine*, 20(4):23–34.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *Proc. IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Edinburgh (UK).

Jörg Hoffmann and Bernhard Nebel. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Alexander Huber and Bernd Ludwig. 2002. A natural language multi-agent system for controlling model trains. In *Proceedings AI, Simulation, and Planning in High Autonomy Systems (AIS 2002)*, pages 145–149, Lissabon.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Department of Linguistics, Göteborg University, Göteborg, Sweden.

Oliver Lemon, Alexander Gruenstein, Alexis Battle, and Stanley Peters. 2002. Multi-tasking and collaborative activities in dialogue systems. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, pages 113–124, Philadelphia.

Henry Lieberman and José Espinosa. 2006. A goal-oriented interface to consumer electronics using planning and commonsene reasoning. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*. ACM, ACM Press.

Bernd Ludwig. 2004. A pragmatics-first approach to the analysis and generation of dialogues. In Susanne Biundo, Rhom Frühwirth, and Günther Palm, editors, *Proc. KI-2004 (27th Annual German Conference on AI (KI-2004)*, pages 82–96, Berlin. Springer.

Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill.

Massimo Poesio and David Traum. 1998. Towards an axiomatisation of dialogue acts. In J. Hulstijn and A. Nijholt, editors, *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 207–222, Enschede.

David R. Traum and Elizabeth A. Hinkelman. 1992. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–592.

David Traum. 1994. *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. thesis, Computer Science Department, University of Rochester.

Alexander Yates, Oren Etzioni, and Daniel Weld. 2003. A reliable natural language interface to household appliances. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 189–196. ACM, ACM Press.

Claus Zinn. 2004. Flexible dialogue management in natural-language enhanced tutoring. In *Konvens 2004 Workshop on Advanced Topics in Modeling Natural Language Dialog*, pages 28–35, Viena, September.