Meta-reasoning for intelligent dialog repair

Khemdut Purang and David Traum and Darsana V Purushothaman and Waiyian Chong and Yoshi Okamoto and Don Perlis

November 9, 2000

Paper ID: NAACL-2001-0087

Keywords: meta-reasoning, dialog, logic, reference, contradiction

Contact Author: kpurang@cs.umd.edu

Under consideration for other conferences (specify)?

Abstract

We present a logical approach and implmentation of a real-time rational agent for dialogue management, which includes an ability to represent, reason about, and repair (both implicitly and explicitly) incoherence in dialogue. We exemplify a target problem of negative feedback in action directive subdialogues, with an ability to make reference resolution sensitive to dialogue context and like intentions. The resulting system is able to behave more sensibly in several situations than earlier dialogue management systems in these same conditions.

Meta-reasoning for intelligent dialog repair

Abstract

We present a logical approach and implmentation of a real-time rational agent for dialogue management, which includes an ability to represent, reason about, and repair (both implicitly and explicitly) incoherence in dialogue. We exemplify a target problem of negative feedback in action directive subdialogues, with an ability to make reference resolution sensitive to dialogue context and like intentions. The resulting system is able to behave more sensibly in several situations than earlier dialogue management systems in these same conditions.

1 Introduction

Miscommunication between participants in a dialogue is a very common phenomenon, even more common in human-computer dialogue than that between humans, because of the more limited speech recognition and language capabilities (Perlis et al., 1998; Sadek and De Mori, 1998; Zollo, 1999). It is thus crucial for effective dialogue systems to have useful strategies for dealing with miscommunication in flexible and efficient ways. A large part of the problem is an ability to recognize when such miscommunication problems occur, and abilities to correct the problem, both non-intrusively, by correcting an internal representation or state to be aligned with the user, or, more intrusively, by asking for the appropriate kind of feedback or repair. Having an explicit representation of the dialogue state, and logical reasoning involving this representation (e.g., (Bretier and Sadek, 1996; McRoy et al., 1997; Traum and Andersen, 1999)), can help expand the range of miscommunication that can be recognized. Adding metareasoning about the representations can allow further kinds of repair strategies, explicitly referring to some aspect of the dialogue or the intentional states related to it.

In this paper, we present a real-time logical approach to miscommunication detection and repair, specifically in the context of negative feedback follow-ups to responses to action requests, such as those corresponding to the following sub-dialogue (where X and Y might be the same):

```
User: Do X
System: [does something]
User: No, do Y
```

The simplest of such cases, in which the user just changes her mind or corrects a recognition or performance error, are within the capability of existing dialogue systems, such as (Ferguson et al., 1996). We will also consider more complex cases, however, in which one must recognize the (seeming) incoherence of the third utterance (`do Y`), in order to repair the misunderstanding, either intrusively or non-intrusively. Previous systems are mostly oblivious to this kind of phenomenon, handling incoherent follow-ups similarly to normal rejections. Moreover, most previous approaches that make use of (in)coherence to derive alternative interpretations are not able to also handle the cases of miscommunication or change in intention.

In the next section, we discuss concrete examples of the above type of sub-dialogue. In section 3 we discuss our approach to dialogue management, using a time-sensitive logical reasoning agent that is sensitive to inconsistency. In section 4, we give more details of the implementation, continuing in section 5 with more details about how the approach has been implemented to address the examples. We then conclude with related and future work.

2 Motivating examples

The kinds of action-request subdialogues can occur in many contexts in which a system can perform actions for a user. However, for concreteness, we give examples from the TRAINS-96 (Ferguson et al., 1996) domain, in which a user interacts with a dialog agent to plan train routes. As illustrative of the general issue of action subdialogues, we will first consider a few scenarios in this domain in which there are two relevant trains – Metroliner and Bullet – that are related to Toronto (e.g., one started the planning scenario there, and one is there now). A mention of *"the Toronto train"* is thus ambiguous and could refer to either Metroliner or Bullet. Incorrect resolution of this ambiguity can be a factor in miscommunication, for instance in the following four cases:

1 User: Send the Toronto train to Montreal
System:[*sends Metroliner to Montreal*]
User: No, send the Toronto train to Syracuse

2 User: Send the Toronto train to Montreal
System:[*sends Metroliner to Montreal*]
User: No, send Bullet to Montreal

3 User: Send the Toronto train to Montreal
System:[*sends Metroliner to Montreal*]
User: No, send the Toronto train to Montreal

4 User: Send the Toronto train to Montreal
System:[*sends Metroliner to Montreal*]
User: No, send the Toronto train to Montreal
System:[*sends Bullet to Montreal*]
User: No, send the Toronto train to Montreal

In the first scenario, it seems that the user, by requesting Syracuse instead of Montreal, is modifying his own original intention and thereby correcting his original request. The system should be able to recognize that the user has changed his intention, take appropriate action to undo the effect of the first utterance and then proceed to fulfill the new intention of the user (as is done also by the Trains-96 system).

In scenario 2, a likely diagnosis is that the system's interpretation of *"the Toronto train"* to mean Metroliner is erroneous and hence the user corrects to Bullet. It is also possible that the user changed his mind and would now like Bullet to be sent instead of Metroliner. In either case, the system should realize that the action just executed is not what the user desires, and then undo the effects of that action before proceeding to send Bullet.

Scenario 3 is similar to scenario 2 from the point of view of the user's intention, which is to send Bullet and not Metroliner to Montreal. However, in 2 the name "Bullet" is used as a disambiguating correction, whereas in 3, the user repeats the same expression "*the Toronto train*". To correctly implement the correction, the system must recognize that the user rejected the system's initial interpretation of *"the Toronto train"* as Metroliner, reinterpret it as Bullet, and then send Bullet to Montreal. This is the scenario that we will focus on in some detail in this paper. Note that, on the surface, it is very similar to Scenario 1, however the system must take care to interpret "*the Toronto train*" differently to avoid an undesirable looping condition.

Scenario 4 is like scenario 3 but this time the user rejects the move of Bullet to Montreal. In this case, the system considered two trains as being the possible referents of "the Toronto train" and neither of them is accepted by the user. The system can then initiate a clarification by asking the user to specify the name of the train he wants to send.

The interesting issue illustrated by these scenarios is the interplay of context with the user intentions and interpretations of the utterances. The interpretation of the corrective request cannot be done in isolation from the context in which the rejection occurred. Conversely, the context of an utterance is partly determined by the interpretations and user intentions of prior utterances. Hence, a conversational agent that can communicate efficiently in all the given scenarios should have the ability to represent and reason about the interpretations of and intentions behind a user's utterance as well as the changing context. Furthermore it must be able to reason about the reasoning it did to get these interpretations and intentions when things go wrong.

3 Approach

Our approach towards the problem of implementing a conversational agent capable of engaging in the kinds of subdialogues illustrated in the previous section is to axiomatize the behavior declaratively in a logic and to use inference in that logic to generate the behavior of the agent in real time. The description of the agent should capture all relevant aspects of the behavior, including the detection of miscommunication by the agent and its subsequent response.

3.1 Miscommunication-free dialog

We start with axioms that describe the behavior of the agent in the absence of miscommunication. These axioms model the relations between (1) the utterance of the conversational partner and the agent's view of the intentions of the partner; and (2) these derived intentions of the partner and the resulting response of the system. These axioms represent information about the rules of conversation and the topic of conversation. The logic uses them to compute derivations that give rise to the behavior of the agent.

Since it is not in general possible to know with certainty what the intention of someone is from his utterances, the axioms are defeasible and the agent computes the best consequences of the input given the knowledge it has. If there is no miscommunication, the intentions that the agent attributes to the user will be correct and these sorts of axioms are sufficient for error-free conversations. But error-free conversations are very rare (Perlis et al., 1998) and miscommunication needs to be taken into account.

3.2 Detecting miscommunication

Incoherence in conversations is typically a good signal that miscommunication has occurred. This inco-

herence is intuitively seen as occurring among the interpretation of the utterances and information about the topic of conversation and the rules of conversation. If the representation is appropriate, the feeling of incoherence should manifest itself as inconsistency in the logic.

The presence of inconsistency can therefore signal to the agent that miscommunication might have occurred (inconsistencies can have other sources, for instance if the agent's knowledge base is inconsistent to begin with). An important aspect of tolerating inconsistency is the ability to isolate the inconsistent part of the database so it does not derive more false formulas.

3.3 Repairing miscommunication

The behaviors for treating inconsistency are specified in the axioms describing the agent in just the same way as the behavior of the agent in the absence of miscommunication is specified. The difference here though is that the axioms do not just represent relations among the entities in the object-domain of the logic, but also relations among the very representations in the agent's knowledge base and about the reasoning of the agent itself.

To engage in the dialogues in the previous section, we need the agent to be able to reason about the process of interpreting the utterances of the user, including examining the inference steps that led from the utterance to the interpretation and response of the agent. To do that we first need to represent the derivation of the interpretation from the utterance. The time at which utterances were made and the formulas derived can also be useful in deciding which of conflicting propositions to prefer.

In a logic with metarepresentation and metareasoning abilities, we can specify axioms that will examine the reasoning that led to the inconsistency and correct the errors that may have occurred. The errors may have been the result of bad default choices made in deriving the interpretation. The examination of these derivations can then lead to alternative choices that change the interpretation of the utterance and can solve the miscommunication. Another resolution strategy can be to just ask for clarification.

A similar approach can be used if the agent is explicitly told of the miscommunication by the dialog partner. The information provided is expected to be inconsistent with some formulas in the agent's knowledgebase and that leads to the correction of the agent's views if necessary.

4 Implementation Building Blocks

In this section we present the building blocks used to implement our approach to miscommunication, sketched in the previous section. We start by describing a logic that is suitable for the approach we propose and discuss an implementation of the logic. Some implementation details mitigate the potential efficiency problems associated with using logics. We then present the overall system in which this logic engine is embedded and in which our conversational agent operates.

4.1 Active Logic

For the approach we discussed above to work, we need a logic that has the following properties:

- The logic needs to be situated and executable in time so that observation of a new utterance triggers the appropriate reasoning and results in a response without untimely delay.
- The logic has to tolerate inconsistencies. In first order logics, the presence of an inconsistency allows the derivation of all sentences in the language, which is clearly undesirable. We need to be able to recognize that there has been an inconsistency and prevent the further derivation of false sentences from it.
- To reason about the inconsistency, we will need a logic that allows metareasoning. We need to be able to express relations among formulas in the logic and to examine derivations of these formulas.

These properties are present in active logic (Elgot-Drapkin and Perlis, 1990; Elgot-Drapkin, 1988). Active logic was originally developed to reason about approaching deadlines and represents an attempt to blend inference and computation in the same formalism. Active logic has a notion of the current time and of the current contents of its database. The inferences that are possible at a particular time depend on the formulas present at that time rather than on all possible theorems of the logic. The logic is executed in steps so that the set of formulas at each step is computed from the set of formulas at the previous step through the inference rules. Active logic has been used for deadline-coupled planning (Nirkhe et al., 1997), Gricean pragmatics (Perlis et al., 1996), and reasoning about presuppositions (Gurney et al., 1997) among other things.

The language of active logic is similar to that of a first order logic except for the following:

- There is a predicate fluent $now(t)$ that represents that the step is t at that moment. This allows one to describe behaviors that depend on the time other events happened and to reason about the evolution of the reasoning.

- Formulas in the language can refer to other formulas. This allows one to reason about the reasoning of the logic and therefore about contradictions.
- There is a predicate $contra(\phi, \neg\phi, t)$ that expresses that the database contained both ϕ and $\neg\phi$ at time t. This formula can be used to trigger reasoning about the inconsistency and attempt to rectify it.

Active logic has some inference rules not found in first order logics:

- A central rule is the clock rule which advances the logic's internal record of time:

$$\frac{t \quad : \quad now(t)}{t+1 \quad : \quad now(t+1)}$$

This rule defines the steps: from the fact that it is step t at the current step, the step number in the next step is $t+1$.

- The contradiction detection rule is triggered by the presence of direct contradictions and asserts $contra(\phi, \neg\phi, t)$ so that the fact that there has been a contradiction can be used for further reasoning.

$$\frac{t \quad : \quad \phi, \neg\phi}{t+1 \quad : \quad contra(\phi, \neg\phi, t)}$$

4.1.1 Alma

Alma is an implementation of active logic that is used in our work that includes a number of design decisions extending and making active logic inference efficient. These include the following:

- inheritance of formula (except for now(t), contradictands, and other explicitly dis-inherited formula) is implicit from time step to time step.
- Alma does not compute all the possible inferences at each step but instead does so only for the newly derived formulas at that step. This saves a lot of computation by not repeating all the inferences already done. This also makes it easier to respond just once to an event.
- Alma has three versions of the material conditional ($\rightarrow$) that can be used to control the inference: the usual if, but also and fif and bif, which are used when we know that the implication will only be used in forward or backward chaining respectively. This can lead to efficiency gains.
- On detection of a contradiction, Alma asserts the *contra* predicate and also *distrusts* the contradictands and their consequences. When a formula is distrusted, it still appears in the database so that we can *reason about* it, but it cannot be *used* in further inferences. If while reasoning about a contradiction we decide which contradictand is likely to be correct, we can *reinstate* it.
- In addition to *contra* various other meta-logical predicates are available that relate formulas to their properties.

4.1.2 Carne

There are certain aspects of the behavior of the agent that we may not need to model logically. Alma can farm out these procedural jobs to a separate action processor called *Carne*. The Carne process runs asynchronously with Alma and adds observations resulting from of its computations to Alma when they are available. Carne also serves as an input-output module to Alma, converting the interface language to and from logic.

The line between which parts of the behavior are to be modeled logically and which procedurally is not fixed and depends on the particular application which determines the computations that we need to reason about. This allows us to trade the flexibility of logic for the efficiency of procedures.

4.2 Trains

The Trains-96 system (Ferguson et al., 1996) engages a user in dialogue about train routing, such as the examples in section 2. The Trains-96 system consists of several modules that communicate through a central hub using KQML (Group, 1993). Modules of interest include the parser, the dialogue manager, the problem solver, and the output manager. The problem solver maintains a view of the Trains-96 world and generates plans for moving trains. It can be queried about the state of the world and requested to generate new plans.

We have replaced the Trains-96 discourse manager with *Acdm* (Alma Carne Dialogue Manager), our conversational agent axiomatized in Alma and following the approach described above. Recognized speech or typed input is sent to a parser, which produces a logical form that is sent to Acdm (through Carne's interface), which then interacts with the problem solver to answer questions from the user or to plan routes. The response of Acdm is then sent to the output manager which generates an appropriate natural language output and updates the Trains-96 display. Carne handles the conversion to and from KQML and logic.

5 Our solution

The design of the representations of Acdm follows that proposed in (Traum and Andersen, 1999). Briefly, an action directive exchange has several different levels of representation, corresponding to intermediate stages in the computation of contextual utterance meaning, each of which could be the source of miscommunication. The levels of greatest interest here are: 1. L-req: the actual words said; 2. I-req: the direct logical interpretation preserving

potentially ambiguous indexical and indirect references ambiguities; 3. D-req: the disambiguated version of I-req. In our work, we have slightly modified and extended the design from (Traum and Andersen, 1999). In the following sections, we sketch our solution to the miscommunication problem focusing on some of the representations and axiomatizations of domain information relevant to example 3 in Section 2. Note that in what follows, the axioms and representations have been edited for clarity. The assertions in the database are of the form NAME: FORMULA where NAME is an integer and the formula is written prolog style. The entire reasoning episode that we describe here can be recreated by running a Java applet on our web page (URL not provided yet to preserve the anonymity of the review).

Set up We follow the dialog from a point in which there are two trains in Toronto: Metroliner and Bullet. The user then utters "Send the Toronto train to Montreal".

L-req and I-req The utterance after parsing and translation into logic by Carne is asserted as the L-req and I-req levels in Alma. This results in assertions like the following in the Alma database. The assertions shown here are the representation of "the Toronto train"

```
2841: ireq(class(v11879, train), kqml294)
2837: ireq(at-loc(v11879, v11868), kqml294)
2832: ireq(class(v11868, city), kqml294)
2830: ireq(lex(v11868, toronto), kqml294)
```

The Toronto train is represented as variable `v11879` which is not mapped to any object in the domain but for which we have a description: that it is at a city and that city's name is Toronto.

D-req Resolving the reference and disambiguating "the Toronto train" results in the D-req representation. This occurs in several stages. We start by noticing that there is no specified domain object associated with v11879 (the lex is null):

```
2890: dreq(lex(v11879, null), kqml294)
```

The logic then attempts to find the referent of this object. The strategy used depends on the type of utterance. For the current case, this results in a query to the domain problem solver to find the trains at Toronto.

The problem solver reply when processed results in two candidates for "the Toronto train":

```
2985: dreq(candidates(v11879, [metroliner,
        bullet]), kqml294)
```

To choose between the candidates the system looks for the first candidate that meets an additional intention-based criterion of not being proscribed from moving to the goal (in this case Montreal). To start with, there are no such proscriptions in the database, and in this case, the choice is arbitrarily Metroliner:

```
2994: dreq(lex(v11879, [metroliner]),
          kqml294)
```

Intention The intention of the user is then computed as:

```
2997: move([metroliner], montreal)
```

Response Next, the domain plan reasoner is again consulted, to find a plan that will send Metroliner along train tracks to reach Montreal. When such a plan is found, it is then executed, updating both the plan reasoner's internal database (so it can respond appropriately to future location and plan requests) as well as multi-modal feedback to the user, including drawing the route on the display, and verbally telling the user of the chosen path and movement. For brevity, we skip the logical and inferential details.

Rejection The user then says "No". As before, representations at the multiple levels are computed. Here we assume that the "No" refers to the last action of the system so the intention of the user now is seen as:

```
3448: not(move([metroliner], montreal)
```

This contradicts formula 2997 above and this is noticed by Alma at step 1494 and we get:

```
3450: contra(3448, 2997, 1494)
3452: distrusted(3448, 1494)
3451: distrusted(2997, 1494)
```

Note that the `contra` predicate has as arguments the names of the formulas, as well as the time step at which the contradiction was detected.

The contradiction is simply resolved in this case by preferring the later formula, using the axiom below. Similar axioms are provided for the other possibilities.

```
contra(X,Y,Z) & name_to_formula(X,Form) &
  Form = [move(Engine,City)] &
  name_to_time(X,T1) & name_to_time(Y,T2) &
  T2 > T1
-> reinstate(Form)
```

This results in a new assertion of

```
3455: not(move([metroliner], montreal))
```

and the action carried out by problem solver and output manager is to undo the last move.

Correction Now the user repeats "Send the Toronto train to Montreal". Once again the same

process as above occurs. In this case too, when disambiguating "the Toronto train" we have two choices: Bullet and Metroliner. But since Metroliner has been proscribed from moving to Montreal, in the cancellation above, the disambiguation procedure described above picks Bullet:

```
3934: dreq(lex(v11992, [bullet]), kqml416)
```

This then results in Bullet being sent to Montreal, as the user intended.

If the user rejects that too and again repeats the original request, as in example 4 above, there will be no unrejected possibilities left and then Acdm engages in a dialog with the user to find out exactly which train is to be sent.

This behavior contrasts with that of the Rochester Trains-96 system where the system will keep sending Metroliner (or whichever engine its separate reference resolution component selects) to Montreal. It never realizes that there is a miscommunication and therefore can't correct it, seeing each cancellation and action directive as sequential changes of the user's plan, regardless of the lack of coherence. A major difference here is the deliberation Acdm does about its own reasoning.

6 Related Work

One main strand of approaches to dialogue management builds on the view of dialogue as a rational, cooperative form of interaction among agents.[1] It is assumed that the maintenance of correct interpretation context is a mutual goal of the participants (Cohen and Levesque, 1991). Building on this basic assumption, some notion of coherence is defined to identify and explain misunderstanding between the participants of a dialogue. Intention usually plays a major role in reasoning about the expectations, as well as formulating repairs in the face of misunderstandings.

6.1 Coherence

Central to the analysis of dialogue is the notion of *coherence*. Miscommunication is assumed when this notion of coherence is violated. Most people would agree that coherence means the lack of contradiction. However, some more aggressive models of dialogue have based their notion of coherence on the expected behavior of agents in conversation. An action is explained as a manifestation of misunderstanding if it is not "expected", given the prior interaction (McRoy, 1998).

[1] See (Cohen, 1996; Sadek and De Mori, 1998) for comparisons between this and other approaches.

Sacks et al. (1974) introduce *adjacency pairs* to model the expected continuations of an interaction: adjacency pairs are sequences of speech acts (e.g., question-answer pairs) such that, after the first element occurs, the second one is expected. However, agent behavior can not directly be explained by means of such strict interactional rules (Levinson, 1981).

To relax such rigid restriction, more sophisticated models have been introduced. In McRoy and Hirst (1995), Traum and Hinkelman (1992), Traum and Allen (1994), the speech acts occurring in the last conversational turn, together with the existing dialogue context, are used to predict which speech acts the interlocutor should perform if the interaction goes well; a deviance from the expected behavior is taken as a sign that some interaction problem is occurring and the presence of a misunderstanding is hypothesized.

6.2 Intention

Another powerful instrument in the analysis of dialogue is intention. Dialogue can be analyzed from the intention recognition point of view (Cohen et al., 1981): when an agent acts, a relation of his action with the interaction context is consulted to see whether the action represents an attempt to satisfy any intention expressed by the partner, or can be inferred from the partner's plans, or is a further step in a plan that has already started.

The intention approach to dialogue interpretation enables a more flexible notion of coherence: an utterance is considered coherent as long as certain relations can be identified among its underlying intentions and those of the dialogue participants. For instance, Ardissono et al. (1998) consider an utterance coherent with the previous context if and only if it can be interpreted as a means for the speaker to achieve an unsatisfied goal which realizes one of the pre-defined *coherence relations*.

6.3 Detection and Repair

Much work on analysis of dialogue is predominantly plan-based (Allen, 1983; Litman and Allen, 1987; Carberry, 1990; Grosz and Sidner, 1990; Lochbaum, 1994). Under the cooperative assumption, every turn in a dialogue is seen as an act to jointly carry on some mutual goals of the participants. Interpretations are recognized by identifying a plan-based relation (such as subgoal, precondition) linking the utterance and the previous context. The absence of this relation is taken as a sign of lack of coherence. In some cases, the notion of coherence is based on the expectation formed by using the dialogue con-

text and other information such as the speech acts in the last utterance. McRoy and Hirst (1995) build on this approach and introduce metaplans to diagnose misunderstandings and formulate repairs when the expected behavior is violated. However, their metaplans only analyze the surface expectations introduced by performing a speech act; the absence of a deeper intentional analysis limits their approach to the treatment of misunderstandings on speech acts. Ardissono et al.(1998) extend the plan-recognition algorithm to deal with misunderstandings on domain level actions also. In their model, when misunderstandings are recognized, a meta-level action results in a subgoal being posted to resolve the misunderstanding; repairs are actions that are performed to satisfy these goals.

McRoy and Hirst (1995) also provide a computational theory of coherence in dialog that accounts for factors such as social expectations, linguistic information and mental attitudes such as belief, desire, or intention. According to this theory, an action is consistent with a discourse only if the beliefs that it expresses are consistent with the beliefs expressed by prior actions. Together, these different factors enable a system to distinguish between actions that are reasonable, although not fully expected (such as an incorrect answer or a clarification question) from actions that are indications of misunderstanding. A weakness, however, is it does not allow for straightforward change in intention, such as our example (1), in Section 2.

In all the models described above, the more aggressive notion of coherence based on expectation is adopted. Not coincidentally, these models usually break down in the face of topic shift or change of subject. The existence of expectation presupposes, explicitly or implicitly, some sort of schema on how the dialogue would progress. For instance, in plan-based approaches, these schema are embodied in the pre- and post- conditions of the action schema, while McRoy and Hirst's (1995) model includes explicit schema to help the interpretation of dialogue exchanges. Such knowledge is domain dependent, and it poses a limit to the applicability of the model to general domains. Ardissono et al. (1998) argue that it is a basic component of any system aiming at deep interpretation of dialogue; but the point is whether this information can be incorporated conveniently and dynamically. With an eye to *conversational adequacy* (Perlis et al., 1998), we need to let open the possibility of adding this information dynamically during the conversation, with perhaps help from the user. By adopting a logical framework and using metareasoning to perform the recognition and repair dynamically, we keep hard-coded knowledge in our system to a minimum, thus retaining maximal flexibility, also allowing both real corrections of intention, and incoherence-based system-initiated repair.

7 Future Work

Unlike traditional Artificial Intelligence systems, Alma is an inference engine that incorporates a history of its reasoning as it runs. We believe that this unique, flexible, feature yields a powerful contribution to our goal of building a dialogue agent that is capable of "common sense reasoning" so that communication between a user and the agent respects the *co-operative principle* (Grice, 1978; Grice, 1975):

> *Make your contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk in exchange in which you are engaged.*

To further this goal, we will design future versions of our agent to have the following features. The agent should make an assumption that the user's utterance is co-operative, and its "common sense reasoning" will verify whether the utterance is conversationally adequate. Inference procedures will include lexical and semantic disambiguation. The agent should be able to understand spelling in meta-dialogue, for instance. It should also be able to make definition queries if necessary. Referential expressions have to be appropriately interpreted by the agent. Linguistic theories provide a set of candidates to which the expressions can make reference. However, some candidates can be eliminated if the agent is able to reason about a discourse domain and to learn about its relationship to the world. We are currently investigating issues on how this selection/inference process takes place, and how the agent can learn the process. We are collecting and analyzing empirical data on definite expressions in order for us to better understand the process. We strive to work on these issues so that the agent is capable of making even better inferences from both semantic/logical meanings and conversational principles.

References

James [F.] Allen. 1983. Recognizing intentions from natural language utterances. In Michael Brady and Robert C. Berwick, editors, *Computational Models of Discourse*. MIT Press.

Lilian Ardissono, Guido Boella, and Rossana Damiano. 1998. A plan-based model of misunderstandings in co-operative dialogue. *International Journal of Human-Computer Studies/Knowledge Acquisition*.

P. Bretier and M. D. Sadek. 1996. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III — Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg.

S. Carberry. 1990. *Plan Recognition in Natural Language Dialogue*. The MIT Press, Cambridge, MA.

Phillip R. Cohen and Hector J. Levesque. 1991. Confirmations and joint action. In *Proceedings IJCAI-91*, pages 951–957.

P.R. Cohen, C.R. Perrault, and J.F. Allen. 1981. Beyond question answering. In W. Lehnert and M. Ringle, editors, *Strategies for Natural Language Processing*, pages 245–274. Lawrence Erlbaum, Hillsdale.

Phil Cohen. 1996. Dialogue modeling. In Ron Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue, editors, *Survey of the State of the Art of Human Language Technology*, chapter 6.3. Cambridge University Press, Cambridge, MA.

J. Elgot-Drapkin and D. Perlis. 1990. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98.

J. Elgot-Drapkin. 1988. *Step-logic: Reasoning Situated in Time*. Ph.D. thesis, Department of Computer Science, University of Maryland, College Park, Maryland.

George M. Ferguson, James F. Allen, Brad W. Miller, and Eric K. Ringger. 1996. The design and implementation of the TRAINS-96 system: A prototype mixed-initiative planning assistant. TRAINS Technical Note 96-5, University of Rochester.

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and semantics 3: Speech Acts*, pages 41–58. Academic Press.

H. P. Grice. 1978. Further notes on logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and semantics 9: pragmatics*, pages 113–128. Academic Press.

Barbara J. Grosz and Candace L. Sidner. 1990. Plans for discourse. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press.

External Interfaces Working Group. 1993. Draft specification of the kqml agent-communication language. available through the WWW at: http://www.cs.umbc.edu/kqml/papers/.

J. Gurney, D. Perlis, and K. Purang. 1997. Interpreting presuppositions using active logic: From contexts to utterances. *Computational Intelligence*, 13(3):391–413.

S.C. Levinson. 1981. The essential inadequacies of speech act models of dialogue. In M. Parret, M. Sbisa, and J. Verschueren, editors, *Possibilities and Limitations of Pragmatics*, pages 473–492. Benjamins, Amsterdam.

D. J. Litman and J. F. Allen. 1987. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200.

Karen E. Lochbaum. 1994. *Using Collaborative Plans to Model the Intentional Structure of Discourse*. Ph.d. dissertation, computer science department, Harvard University, Cambridge, MA.

Susan W. McRoy and Graeme Hirst. 1995. The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4):5–478.

Susan W. McRoy, Susan Haller, and Syed Ali. 1997. Uniform knowledge representation for language processing in the b2 system. *Journal of Natural Language Engineering*, 3(2/3):123–145.

Susan McRoy. 1998. Achieving robust human-computer communication. *International Journal of Human-Computer Studies*, 48:681–704.

M. Nirkhe, S. Kraus, M. Miller, and D. Perlis. 1997. How to (plan to) meet a deadline between *now* and *then*. *Journal of logic computation*, 7(1):109–156.

D. Perlis, J. Gurney, and K. Purang. 1996. Active logic applied to cancellation of Gricean implicature. In *Working notes, AAAI 96 Spring Symposium on Computational Implicature*. AAAI.

D. Perlis, K. Purang, and C. Andersen. 1998. Conversational adequacy: Mistakes are the essense. *International Journal of Human Computer Studies*, pages 553–575.

H. Sacks, E. A. Schegloff, and G. Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50:696–735.

David Sadek and Renato De Mori. 1998. Dialogue systems. In R. De Mori, editor, *Spoken Dialogues with Computers*. Academic Press.

David R. Traum and James F. Allen. 1994. Discourse obligations in dialogue processing. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics*, pages 1–8.

D. Traum and C. Andersen. 1999. Representations of dialogue state for domain and task independent meta-dialogue. In *Proceedings of the IJCAI99 workshop: Knowledge And Reasoning in Practical Dialogue Systems*, pages 113–120.

D. Traum and E. Hinkelman. 1992. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599.

Teresa Zollo. 1999. A study of human dialogue strategies in the presence of speech recognition errors. In *Working Notes AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, November.