Collaborative Problem Solving in Dialogue Models for Practical Purposes

Renjini Narendranath

Dept. of Computational Linguistics, Universität des Saarlandes
renjinin@coli.uni-sb.de

April 5, 2004

Abstract

In this term paper we study the dialogue modeling issues of collaborative problem solving. We analyze the architecture of Collagen, a middle ware agent based on the collaborative discourse theory. We also discuss specific agents based on Collagen like PACO (intelligent tutoring agent). Collaborative agents based on multi-tasking is another aspect that we study in this paper. A specific agent WITAS, an unmanned vehicle, based on concept of multi-tasking is also discussed.

1 Introduction

In an effort to build intelligent agent based user interfaces, one question that gets addressed quite naturally is, what are the parameters of intelligence for such an agent. An agent is considered intelligent, if it is able to collaborate with human beings to achieve a common goal or task. During such collaborations, the actions of the agent should resemble to human behavior as much as possible. Such agents should be flexible and adaptable to the extent of being user friendly even to a naive user. Dialogue managers form one of the most important modules of such intelligent agents.

In this term paper, we discuss recent works related to the collaborative problem solving aspects of dialogue models. There are three main parts in this paper. First we study and evaluate the architecture of collaborative middleware agent COLLAGEN [CCN01]. Then we go on to study specific agents that are based on Collagen like PACO [RLR+02], VCR agent etc. In the last part, we discuss and evaluate the concept of multi-tasking and collaborative activities of the WITAS [LGP98] unmanned vehicle.

1.1 Organization of the Paper

In Section 2 we explain some basic terminologies. In Section 3 we motivate the concept of collaboration in Human-Computer interaction and introduce the theory of collaboration. The architecture of Collagen is discussed in Section 4 which is followed by a collaborative intelligent tutor based on Collagen in Section 5. In Section 6 we discuss the multi-tasking and collaborative activities in dialogue systems followed by architecture of WITAS unmanned areal vehicle multi-tasking dialogue system, in Section 7. The figures 1, 2 and 3 are borrowed from the paper [CCN01]. The figure 4 and the figure 5 are borrowed from the papers [RLR+02] and [LGP98] respectively.

2 Basic Terminologies

In this section, we discuss some of the basic terminologies related to Dialogues and their computer models.

- *Intelligent Agents*: Generally referred to computer simulated beings who simulate human behaviors. They communicate with human beings using natural language dialogues.
- *Human-Computer Interaction*: The process by which human beings and computer collaborate to achieve a common goal or task.
- *Graphical User Interface (GUI)*: A computer interface which acts as a communication door between human and computer.
- *Multi-tasking*: Handling multiple tasks at the same time.
- *Interleaved Dialogues*: A dialogue in which many conversational threads intervene.
- *Mixed-initiative*: This is the process by which dialogue participants take part in collaborative activities and give suggestions such that an active take part of the participants occurs.

3 Collaboration and its need in Human-Computer Interaction (HCI)

Collaboration is the event in which more than one individual participate and work their way towards a common goal or task. They solve problems jointly through discussions and dialogues, and initiate new actions or suggestions such that it help towards the progression of the task. The event of human to human collaboration and joint actions can be mapped to human computer interaction such that a new genre of collaborative dialogue (software)agents

evolve. These agents are intelligent, robust and are capable handling dialogues with human beings in natural language. Towards this end, one need to address two questions: What are the constituents of a human dialogue which needs to be modeled into an agent? What does it mean for the agent be collaborative in doing joint conversations/activities with human beings?

A conversational agent functions in three basic modules [JGA01]. Interpretation, behavior and generation. Given an utterance, the system should be capable of interpreting the utterance beyond the surface structure of the words. That is, it should be capable of reading between the lines and derive the "other meaning" if any, in addition to the normal meanings of the utterance (also known as communicative intentions). Once the intentions are inferred, the next step for the agent is to decide upon what to do/say next (behavior) and to formulate its own intentions and beliefs. Once this is done, the system needs to utter sentences or perform actions in response to an utterance from the user (generation).

An intelligent user interface is an interface through which the user communicates with the agent and gets relevant information. Such interfaces can be successfully employed in places where information is delivered. For example, Train timings enquiry system. Not only can such agents be employed in give-away information domains, but also to teach the user to perform a task. Such tutors are called intelligent tutors who can teach tasks with lots of dialogues and discussions with the student. Further down in this paper we will be discussing one such tutoring agent called PACO who teaches a task of operating engines in a ship. The agent PACO is designed with Collagen [CCN01], which is one such dialogue management module, integrated into it.

Many of the recent successful dialogue systems have been built with interfaces using WIMP (Windows, Icons, Menus and Pointers), for the simple reason that they are then very user friendly. Collagen also uses WIMP to build graphical user interfaces, so as to communicate with the user. This makes the agent more adaptive and user friendly. The VCR agent shown in Figure 1 is an example of an interface window where one can see the agent, the user and the event (or task) they jointly manipulate. The VCR agent is a simple agent with the task of setting up a VCR, so that the user can plan recording programs with the help of the agent in his absence.

3.1 Collaborative Discourse Theory and Collagen

For modeling a task-oriented discourse structure for collaboration, Grosz and Sidner [BC86] proposed a tripartriate structure that basically models how the mutual intentions and beliefs of the participants of such collaborative discourse accumulate. It consists of an *Intensional structure*, which records the beliefs and intentions of the speaker, *Attentional structure* which maintains a stack of themes in focus and *Linguistic structure* which hierarchically

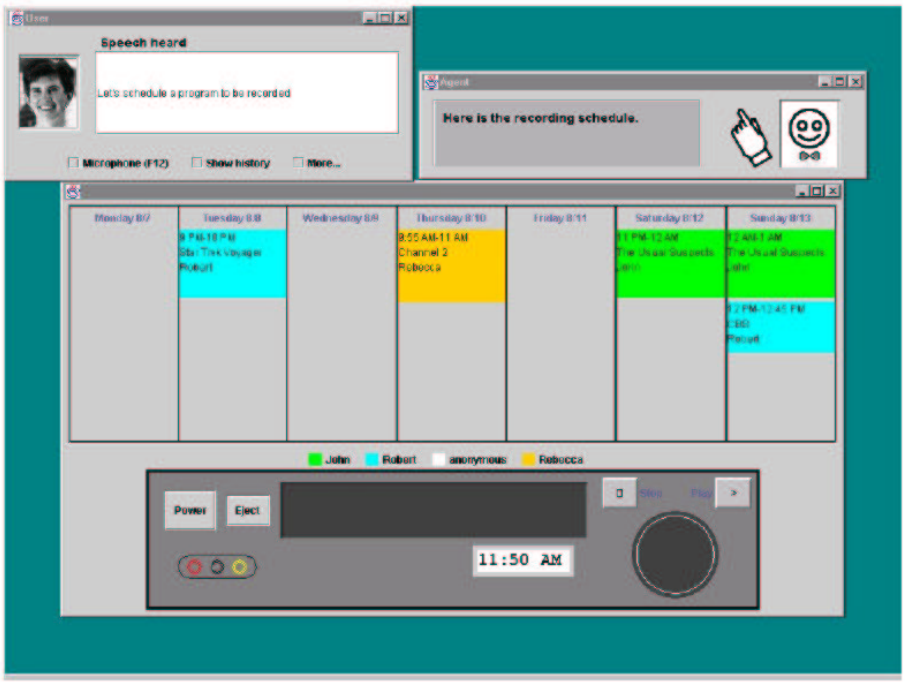

Figure 1: The VCR agent.

keeps track of the exact utterances or actions produced by the conversants. SharedPlans [BS96] is a later formalization is a partial representation of the Intensional structure, and this was supplemented with Lochbaum's intention recognition algorithm [K.E98] which more or less hints at what a conversant means by an utterance.

3.2 Discourse State of the VCR agent

Figure 2 illustrates the partial representation of discourse state of the VCR agent. Here, the *Focus Stack* represents the attentional structure. The plan tree captures the order of execution of plans which is specified by a recipe. A recipe is a set of instructions in order to execute a plan. In plan tree, actions are hierarchically ordered. The node 1 in the plan tree below corresponds to the utterance 1 in the *Segmented History* seen right to the plan tree. When the node 1 action is done successfully, then the agent proceeds to the next item in the plan tree i.e, "Displaying Schedule". This node has to sub tasks: 2 and 3 which is again described in the actions 2 and 3 of the segmented history. Once this step is successfully executed, then the agent proceeds to the next step of expecting the user to add the program details he wishes to be recorded and report any conflict or difference of opinion at this point.

If the user changed his mind and says that he wishes to perform a different task, then in order to perform the task, the intention recognition

algorithm interprets what the intention behind the users utterance is. Then it queries the recipe library, and gets hold of a suitable recipe in order to accomplish this task. This recipe contains a set of items that has to be fulfilled in order to accomplish the task. Then the recipe is decomposed, and the plan tree is made so that each item on the node can be finished in the order. Each node in the plan tree is loaded or stacked on to the focus stack. The top most item of the stack is the "current purpose of the discourse", i.e, item that is next to be executed. After execution, that item is popped off the stack and the item below now becomes the next item to be executed. This process runs recursively till all the elements on the focus stack is worked out successfully. The focus stack mechanism is comparable to the *Question under discussion* module in the GoDiS system [PRES99] developed at Gothenburg University.

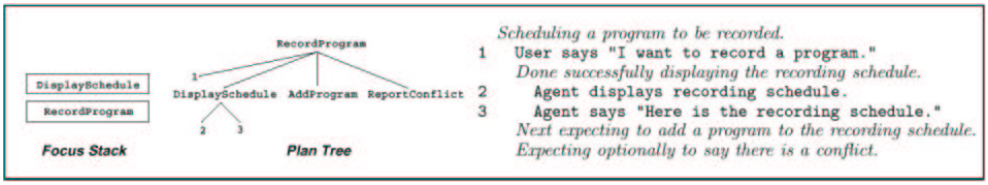

Figure 2: The discourse state of the VCR agent.

A special note to the segmented history: In an elaborate discussion on the "history based transformations" in collaborative discourse [CC98], Rich and Sidner rightly argues that these discourse segments are the very foundation of discourse theory. Each event of the discourse is annotated, intended and recorded in this segmented history. This contiguous list of what has been uttered till now serves as a context or history of events that has been occurred. This was initially conceptualized as a log of all the utterances by the user as well as the agent, so that in case of problem occurance they can backtrace and find out where the problem had occurred. In another application of an air travel agent (which was a non-commercial test bed for the development of Collagen) [CC98] the segmented history was provided with an extra window within the GUI such that all the utterances are listed in this window from the beginning of the conversation onwards. This proved to be useful to the users to visualize what is going on in and to understand if something has gone wrong.

They further argue that the very nature of human interaction in a discourse are of hierarchical in structure, which in some sense is disputable. Natural human dialogues are interleaved in nature than hierarchical. Not always are human dialogues stimulus-response sort of pairs in free variations. But they at times introduces discourse entities which sometimes do not directly refer to the existing set of entities in the context. For example, the phenomenon of lampooning in human dialogues, which is for instance

not meant as a direct answer to a stimuli but a sort of response (negative) in order to stop the opposite conversant from making more comments. The dialogue systems that we study in this term paper are far less complex to handle these kinds of language phenomenon.

4 Architecture of Collagen

In figure 3, we can see the complete modules of a collaborative system built with Collagen. After each utterance produced by the user or the agent, the discourse state gets updated using the Lochbaum's discourse interpretation algorithm, which is the discourse interpretation module in figure 3. This module also contains extensions to include plan recognition [HJ86] mechanism, which infers intentions from actions of the discourse participants. This is a very desirable feature for a dialogue system, in the sense that it plays a key role in successful human-to-human communication where one understands the intention and plans of the speaker and makes appropriate decisions on what to say next. But, the recognition of plan problem is known to be exponential in worst case [H.A90]. Therefore, they have implemented only minimal properties of plan recognition in the Collagen, such as, providing help or clarification to the user. Once the interpretation of the utterance is done the discourse interpretation algorithm updates the discourse state, which causes the discourse generation module to issue a new agenda. The agent executes the entries one by one in the agenda which can be either an utterance or a manipulation of the GUI, and this is now a new input to the discourse interpretation module.

This architecture allows mixed initiative, i.e., the user can at any point of time interrupt such that the agent then responds to this action before continuing further. This is a good option, but to be used with discretion. For instance, this option could lead the system to run into infinite loops if the architecture does not ensure cutting short the unwanted repetition.

4.1 Critical Evaluation of Collagen's Architecture

Let's take the example of the VCR agent. Here, the domain of action for the agent is strictly specified or lets say restricted. The set of actions that the agent can perform are hard-coded into a *library of recipes*. So, depending on the previous utterance by the user, a decision will be made by the system in response. This decision on what to act next will be one among those recipes. This is also one of the limitations of such systems. On the one hand, they are well equipped for a defined domain. But on the other hand, they are incapable for a slightest deviation from its behavior "designed" by the programmer. The recipe libraries can be extended arbitrarily in such a way that all the desirable intelligent qualities can be integrated into such agents. But querying such huge set of recipes in order to find the optimal recipe to

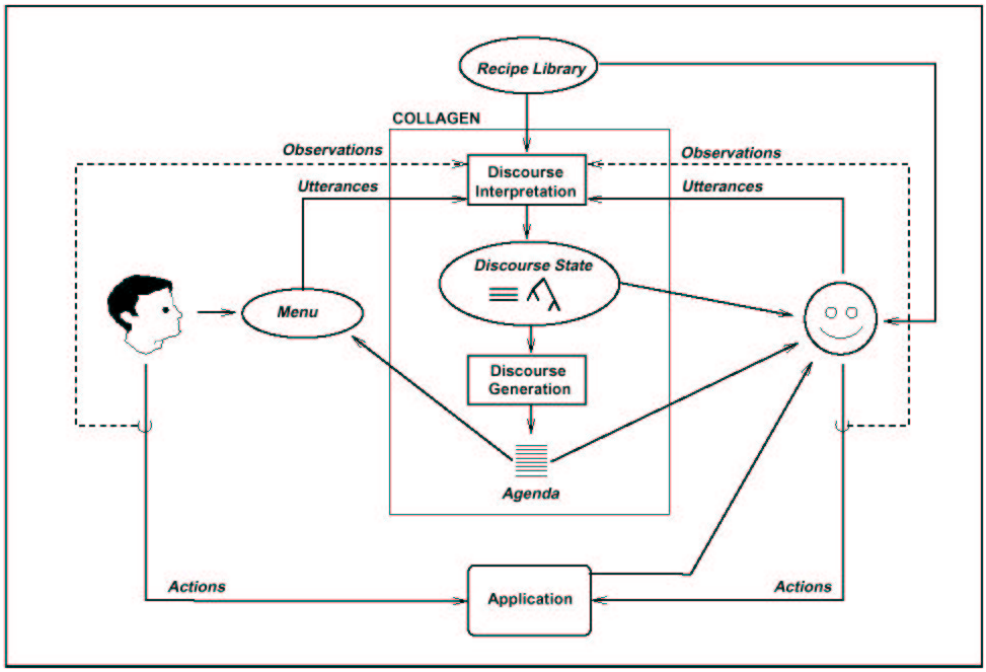

Figure 3: The architecture of a collaborative system built with Collagen.

be chosen given a set of constraints (in the context of what has happened till now) can be mapped into a search problem where the algorithm would have to search in a very large search space. This is particularly a hard problem, if the algorithm is not robust and cannot retract the optimal recipe within a reasonable time.

5 A Collaborative Intelligent Tutor Using Collagen: PACO

PACO or Pedagogical Agent for Collagen is an agent who tutors or helps in constructive learning [RLR+02]. The domain of PACO is teaching procedural tasks using simulated-environment of a Gas-Turbine engine. The methodology followed in the design of PACO is that of mixed-initiative and collaborative problem-solving from the part of student as well as the agent. The agent judges the student's progress from his utterances, and makes inferences based on that. And then it suggests new actions accordingly so that a progressive learning happens.

PACO has the following additional features. It captures the turn taking phenomena of when to hand over the initiative to the student. For example, the dialogue "You take it from here" means that now it is the student's turn to suggest how to proceed further. The near miss plan recognition is additional feature of plan recognition algorithm. This detects the errors done by the student. This feature is good to avoid repetitions, for instance

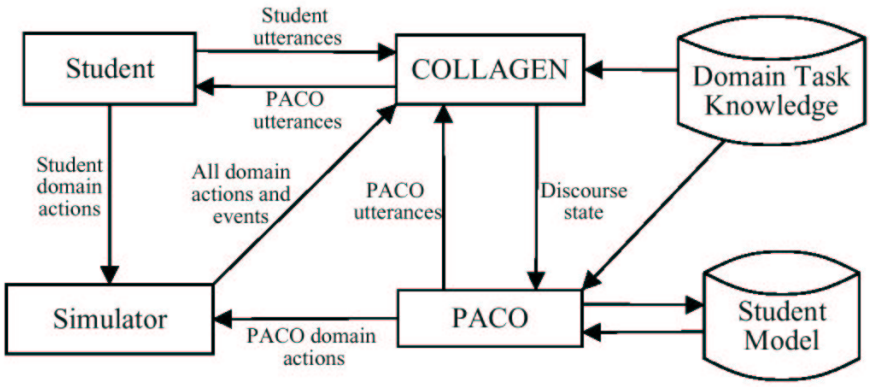

Figure 4: The architecture of PACO.

the student repeats a step which he has previously been done. This also avoids the violation of ordering constraints (i.e, the order in which the sub-activities are to be executed) in the recipes. The problem with this feature is that it thereby makes the ordering constraint too rigid.

5.1 Evaluation on the Degree of Collaboration of PACO

There are many places where PACO's feedback would be "I wasn't expecting you to do it now". This feedback results from the ordering constraint violation by the student. Instead of making clear why such an action is unsolicited, PACO just says that this action is not allowed. This behavior is too rigid and offers very less help to the understanding of the student on why such an action is not called for. This behavior renders a finite state like discourse move. Before venturing out with the system the student is pre-supposed to have prior knowledge of PACO and how it works. This means that a naive user cannot use PACO without being equipped with some initial knowledge of how to operate PACO.

6 Multi-tasking and Collaborative Activities In Dialogue Systems

In this section we address the concept of multiple and concurrent tasks and collaborative activities in dialogue systems as presented in [LGP98]. This work focuses on the aspects of systems which could handle dialogues about multiple concurrent tasks in a coherent and natural way. That is, if we consider a theme in dialogue as a conversational thread, and if the dialogue continues further then we can say that the dialogue proceeds on a single conversational thread. But when we look at natural human conversations, many a times it happens so that people speak about a theme, before that

theme is discussed and completed they start of with a new theme, a new conversational thread. That means, now we have two open conversational threads. Take for instance two people on a long drive in the highway speaking about their plans as they reach their destination. While they discuss this, say the driver asks where is the next tanking station. Now, there are two open conversational threads/ themes. That is, the thread of planning their actions when they reach the destination and the other the query about the next tanking station. Now if the second conversational thread is provided with an answer then that thread is closed and the speaker returns to the older conversational thread. This is an example of interleaved dialogues. There can be any number of such threads either closed or opened which can be taken up at any point of time and continued from there on for further discussion. In that case, even a seemingly closed thread is not really closed beyond further discussion but any time openable/ retractable. That means one can visualize dialogue as a collection of many such threads.

If we are to endow a system with multi-tasking qualities, there are four main issues to be considered.

1. Representation of the discourse state which supports multi-tasking.
2. Dialogue management methods, algorithms that supports such free and natural navigation among the threads.
3. The natural generation of messages in multi-tasking and collaborative dialogues.
4. Representation and reasoning among multiple collaborative activities in dialogue

In [LGP98] the authors describes the project WHITAS where they design an unmanned Ariel vehicle (UAV) which is a small robotic helicopter with an on-board planning system developed at the Linköping University, Sweden. When the UAV is air-bourn, it communicates with the operator down somewhere on earth via natural language dialogue system. And the mission goals for this unmanned helicopter are provided by the human operator via CSLI dialogue system developed at the Stanford University, USA. The domain actions performed by the UAV is to fly to a location specified by the operator, search for a vehicle, follow a vehicle, and land. The vision of the helicopter is enabled by sensory cameras.

6.1 The Challenges in building Robot-Human dialogue systems

One of the main problem poses when the robots exists in a dynamic and changing environments. That is, the robot must not only obey to the operators instructions, but also observe his constantly changing environments

and then report back appropriately. The robot must perform multiple activities which may be successful, failure, become canceled or revised by the operator. The operator should be able to interrupt the robot at any point of time. Nevertheless, the robot should be capable of getting back to the previous conversational thread. The robot should be flexible in the sense that even non experts could use the system with out difficulty. The robot should be able to produce adequate feedback yet it should not overload the user with informations. That is, it should be able to filter out the irrelevant information.

7 WITAS UAV: CSLI Dialogue System Architecture

In this section we will see the different modules of the WITAS dialogue system and how they interact with each other.

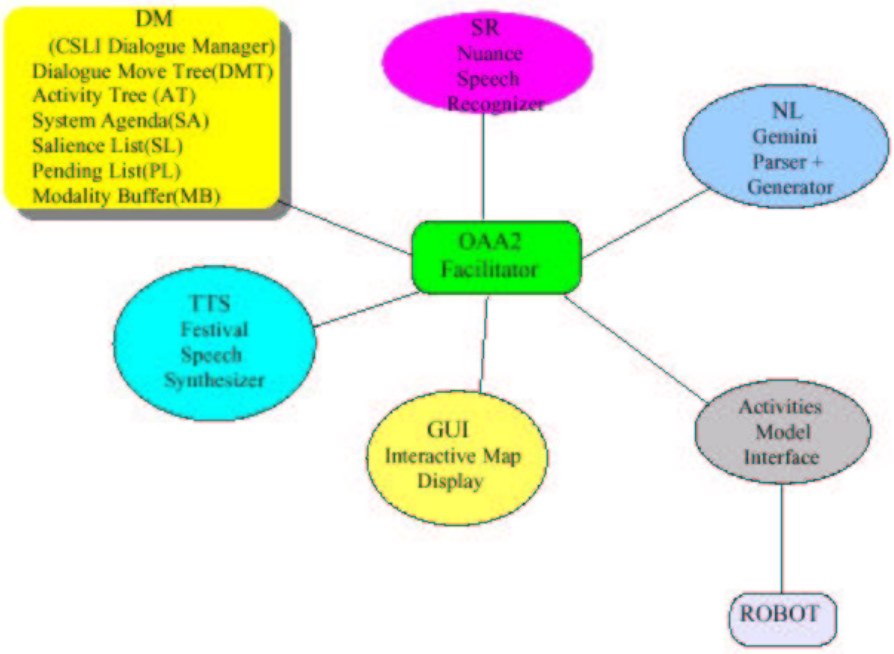

Figure 5: The CSLI Dialogue system architecture.

SR Nuance Speech Recognizer. The incoming human speech is recognized here and it is parsed into a system specific logic form.
NL Gemini Parse and Generator. This module along with the speech recognizer helps in parsing the incoming speech signals. It also helps in generating the logical forms of the sentences that needs to be given for the speech synthesis.
TTS Festival Speech Recognizer. This module gets as input the logical forms of the sentences that needs to be synthesized into natural language sentences.

The GUI interactive map display. This displays the map on which the operator can point to and say for instance "fly here" instead of specifying the name of the place. Similarly the robot can also manipulate the GUI.
OAA2 Facilitator. This facilitates and also restricts the passage of information from one module to another.
Activity Model. There are certain activities that are specific to the UAV. For instance, "locate", "search", "look-for" are some sample activities of the UAV. Activity models specify how to decompose high level commands into sub-activities (same as recipes in Collagen [CCN01]). Then these sub-activities are added as active nodes to the activity tree.
The dialogue move tree (DMT). DMT is a history/ message board of dialogues organized by thread, based on activities. It verifies the incoming utterances based on the context of current discourse state adds to the existing set if it is compatible and rejects if it is not. It has an active node list which controls the order of activities to be performed. The DMT is the place which contains all the threads in the dialogue.
System Agenda. This is the stack or collection of the communicative goals that are to be accomplished.
Pending Lists. This is the list of the questions not answered by the user. These questions are repeatedly asked for some time and if the user chooses not to respond then these questions are removed from the pending list.
Salience List. This list contains an indexical information about the NPs (Noun Phrases) extracted from the logical forms of the sentences from the DMT. Example: "The truck", in the utterance "Follow the truck". This feature helps in the process of echoing. That is, if in the question "the truck" occurs, then in the answer the system can use the anaphora "it" instead of repeating the same NP.
Modality Buffer. This buffer stores the GUI map display inputs (mouse clicks).

7.1 An Example Activity Model

The user says: "We are looking for a truck." The parse of the sentence looks like **(locate, NP[det(a),truck])**. Here "locate" is the activity model which needs to be decomposed into sub-activities. These sub-activities are invoked based on the pre-conditions of the activity. For instance, pre-conditions for the UAV to locate a truck are

1. UAV should be in the air and it should have enough fuel.
2. The engine is running without problem.
3. If the task of locating the truck is already done then skip the whole task and assert this status.

If the pre-conditions holds then proceed further to the next sub-activity. That is, *watch for, follow object, ask complete.* The possible nodes of these sub-tasks in the corresponding activity tree are active, complete, failed, suspended, canceled. These nodes indicate the state of progress of the activity.

7.2 Dialogue Context Model

The dialogue context model addresses two two major issues in dialogue management. One is dialogue modeling/ representation of the discourse and the other is dialogue control algorithm which decides how the information state of the discourse model is updated. The information state update happens with each new input of utterance. The DMT is the function space of he information state update functions. The interpretation and the state update algorithm processes the input conversational move and attach a new node to the active node list of the DMT. If a matching node to the input is already existing, for e.g., adjacency pairs, then the new node gets attached to the existing one. After this attachment the information state is updated eventually.

7.3 Message selection, aggregation and generation

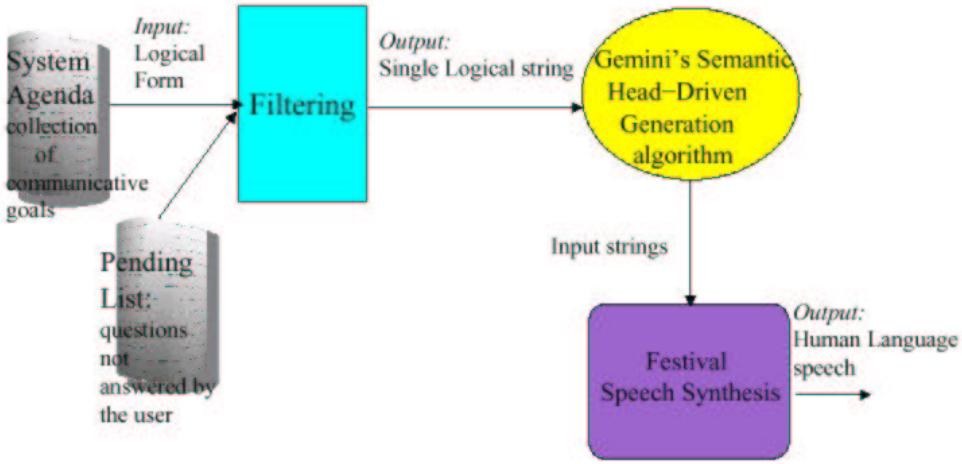

Figure 6: Message generation in CSLI dialogue manager.

The generation of messages is usually difficult when there are many open threads or there exists more than one participants. Some of the bottlenecks in message selection are, for instance, what to say next if the robot moves in a changing environments, progress towards a user specified goal, execution of activities/ tasks, robot's own internal changes, the progress of the

dialogue itself. In a such dynamic and always changing situations a template based approach for message selection is not adequate. We need an extra mechanism which can filter out 'the relevant information', and generate these as natural utterances. The naturality of such utterances can be glorified using anaphoric references, echoing (using the same language-style as the user), variability (not using the same words as the user but slightly changing them). Filters that help in the generation of messages are for instance, relevance filters, recency filters, echoing, variability, aggregation, symmetry and priming, real time message generation etc. As seen in Figure 6, the input logical forms from the system agenda and the pending list gets filtered and the output is given for synthesis of human language speech.

7.4 A Critical Look at the Two Input Stacks: System Agenda and Pending List

The system agenda contains the questions not yet spoken, reports, Wh-questions, warnings and informings. On the other hand, pending lists are of much lesser size compared to system agenda. They contain a list of unanswered questions that have to be repeated over a period of time. The idea of pending list is particularly good because this functions as an independent module and prompts the unanswered questions to the user again, without using the huge dialogue move tree for this purpose. If we were to use the DMT for this purpose it would have incurred unnecessary cost of computation of searching the DMT which contains scores of nodes, and open threads.

7.5 Open Issues with CSLI

The system warnings like "I am out of fuel" would not be affected by any filter. It is not clear in the paper [LGP98], how does the priority gets "prioritized" when more than one such technical warnings occur. A better option might be to check the most important if not all the technical issues in the "Precondition List" which appears in each Activity model of the UAV robot.

References

[BC86] B.J.Grosz and C.L.Sidner. Attention, intensions and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[BS96] B.J.Grosz and S.Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[CC98] C.Rich and C.L.Sidner. Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3/4):315–350, 1998.

[CCN01] C.Rich, C.L.Sidner, and N.Lesh. Collagen: Applying collaborative discourse theory to human-computer collaboration. *AI Magazine*, 22(4):15–25, 2001.

[H.A90] H.A.Kautz. A circumscriptive theory of plan recognition. In *P.R.Cohen, J.L.Morgan and M.E.Pollack (eds.): Intensions and Communication*, pages Chapt.6, 105–133. Cambridge, M.A: MIT Press, 1990.

[HJ86] H.A.Kautz and J.F.Allen. A generalized plan recognition. In *Proceedings of 5th National Conference on Artificial Intelligence*, pages 32–37, 1986.

[JGA01] J.Allen, G.Freguson, and A.Stent. An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, pages 1–8, 2001.

[K.E98] K.E.Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572, 1998.

[LGP98] O. Lemon, A. Gruenstein, and S. Peters. Collaborative activities and multi-tasking in dialogue systems. *Traitement Automatique des Langues (TAL)*, 43(2):131–154, 1998.

[PRES99] P.Bohlin, R.Cooper, E.Engdhal, and S.Larsson. Information states and dialogue move engines. In *Proceedings of the ijcai99 workshop: Knowledge and reasoning in practical dialogue systems*, pages 25–31, 1999.

[RLR+02] J. Rickel, N. Lesh, C. Rich, C.L. Sidner, and A. Gertner. Collaborative discourse theory as a foundation for tutorial dialogue. In *ITS*, pages 542–551, 2002.