A Joint Intention-Based Dialogue Engine

Rajah Annamalai Subramanian[1, 2], Philip Cohen[1]

[1]Natural Interaction Systems, Seattle, USA
[2]Department of Computer Science and Engineering, Oregon Health and Sciences University, USA
rajah@csee.ogi.edu, Phil.Cohen@naturalinteraction.com

ABSTRACT

In this paper we discuss some of the advantages of using a joint intention-based interpreter for building spoken dialogue systems. We describe STAPLE [1], a joint-intention interpreter that enables a system to obtain team and communicative behavior automatically without having to program this behavior explicitly. With this approach there is no necessity for a programmer to indicate when a question should be posed, when information should be shared etc. The interpreter enables the agents to exhibit team-oriented dialogue by interpreting the constructs of Joint Intention Theory(JIT) along with first principles reasoning over a formal semantics of communicative acts [2, 3]. We try to show that STAPLE can subsume and extend the finite-state and frame-based dialogue approaches available today from commercial dialogue systems. In particular, we show how STAPLE can handle *over-answering, dynamic environment changes* and *teamwork* in a general domain independent manner.

Index Terms: joint-intention, speech-acts, collaborative - dialogue, teamwork, multi-agent systems.

1. INTRODUCTION

Many dialogue systems available today are based on either finite-state or frame-based methodology. We briefly introduce these two methodologies, explain some of their short comings and provide our solution for a general dialogue architecture. We compare our research with existing collaborative agent models and the advantages of extending them by using Joint Intention Theory(JIT) .

Most finite-state systems are system-initiated following a predetermined sequence of questions and answers. A sample finite-state dialogue exchange is as follows:

Agent: What month would you like to travel? *User: March*

The agent has a speech recognition grammar that contains all the months in the year. If there is an error in recognition or a failure to interpret the result, an error message is generated. More sophisticated systems can handle responses such as:

User: March will be great! *User: I would like to travel in March please.*

This is a typically done by keyword matching from the utterance. More generally, the finite-state systems tend to have difficulty when the user responds with:

User: March 25th 2006, or *User: 25th, this month.*

These are examples of over-answering, In order to handle such responses, the programmer has to think through all possible ways the user can reply and explicitly program the possible states. For example, to fill three data fields, to capture all possibilities of answering, eight states are required as shown in Figure 1. A "0" indicates unfilled data and "1" indicates that the value is obtained. For example, for a room reservation system, the first, second and third digits could represent the date, the time, and the size respectively of the desired room reservation. Thus "1 1 0" indicates that only the size is missing and thus the system prompts "Please say the size of the room". In general, for n fields, 2^n states are required; this makes it difficult to handle domains with large number of data fields and/or flexible responses.

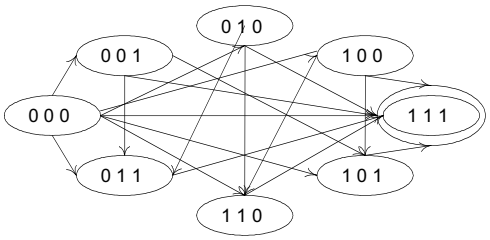

Figure 1: Finite State Dialogue Model [4]

In frame-based systems [4, 5], the system is designed around the task of obtaining missing information. A "frame" is a template made up of a set of typed "slots" that describe those pieces of information that are required by the system. For example, a frame template for a date will embed day, month and year fields. After a query to the user, the system takes as input the user response and the prior frame, and generates as output a subsequent frame that has slots for information that still needs to be obtained. This is done by a set of rules, each having a set of conditions for "firing" based on the user query.

Date & Time & ! Size → Prompt Size Date & ! Time & ! Size → Prompt Time or Prompt Size !Date & Time & Size → Prompt Date

Table 1: Examples Rules from Mercury [4]

The dialogue manager tries to prepare a suitable frame representing its reply to the user, essentially, dynamically deciding the state to which to transition via a set of rules (see Table 1). Frame-based systems are thus similar to finite-state systems, with "dynamic states" generated by the set of pre-defined rules.

These systems generally have trouble understanding "out of frame" responses. The programmer still has to make a decision about all the related data that needs to be combined to form a frame and to follow the process for defining the rules. Another issue with this approach is portability. Some of the frames can be re-used (such as the date frame), but it will usually be necessary to reorganize and create new frames and rules when

moving from one domain to another. Also there is no general solution for handling any dynamic changes that may take place in the environment (Section 3.2).

In contrast to these approaches, we describe how STAPLE [3], a joint intention interpreter, can be used to build collaborative dialogue agents that both generalize and overcome problems with standard finite state and frame based models.

2. THE STAPLE ARCHITECTURE

STAPLE is a multi-stack Belief-Desire-Intention (BDI) [6] interpreter with built-in support for individual and joint goals and intentions. Agents in STAPLE are programmed using the usual Prolog syntax extended by operators for dynamic logic of actions (concurrent actions, test, repetition, etc.), temporal logic (eventually and always), for negation, implication, conjunction, and some other miscellaneous constructs. The STAPLE architecture is as shown in Figure 2.

2.1 Joint Intention Interpreter

Similar to prior work by Grosz and Sidner [7], Joint Intention (JI) theory [8] stipulates what it means for agents to execute actions as a team. This same theory is used to specify formal semantics of communicative acts as an extension of standard speech-act theory [1, 2].

The notion of an agent's commitment to achieving some state in the world is expressed as a persistent goal or PGOAL. An agent x has a persistent goal (PGOAL x p q) if x wants p to become true and cannot give up the goal until it believes that p is accomplished, impossible, or irrelevant (i.e. the relativizing condition q is untrue). The definition of a persistent weak achievement goal or PWAG states that an agent x has a PWAG towards another agent y when the following holds: if agent x believes that p is not currently true then it will have a persistent goal to achieve p, and if it believes p to be either true, or to be impossible, or if it believes the relativizing condition q to be false, then it will have a persistent goal to bring about the corresponding mutual belief with agent y. The notion of teamwork is characterized by joint commitment (also known as joint persistent goal or JPG). The definition of JPG states that the agents mutually believe they have the appropriate goal and that they mutually believe that they each have a PWAG to achieve it (relative to the others' PWAG). A detailed explanation of the JI theory is given in [9, 10]. STAPLE interpreter implements the definitions of JIT (details in [1]) and thus its behavior obeys the theory's predictions.

During the execution of an individual or joint action, an *Intention-Commitment stack* is built in a bottom-up manner, with the bottom element containing the original commitment and the other commitments or sub-goals used to achieve the original commitment layered above it. If one of the goals in the stack becomes true, impossible or irrelevant, all its sub-goals become true, impossible, or irrelevant respectively. However, in the case of a PWAG, new PGOALs are created to obtain the appropriate mutual beliefs. Triggers are created to monitor the escape conditions in the definition of PGOAL (achievement, impossibility or irrelevance), and in the case of PWAGs, mutual belief of those conditions.

One general principle of STAPLE is if it comes across an unbound variable in a term or description for which it requires a value, it generates a PGOAL to know what the referent of that variable is; this is defined as KNOWREF [11].

The Horn-clause *belief reasoner* implements a weak S5 semantics and is capable of reasoning with quantified beliefs within the Horn subset of first-order logic. More information on the belief reasoner can be found in [1].

A *consistency checker* employs the belief reasoner to attempt to ensure that the content of the PGOAL is consistent with the existing commitments and intentions of this agent.

This general agent architecture can be used as a dialogue engine as one of its direct applications. The built-in theory generates appropriate communicative acts and the framework to act as team can naturally fold into a human-computer dialogue to jointly perform a plan or action together.

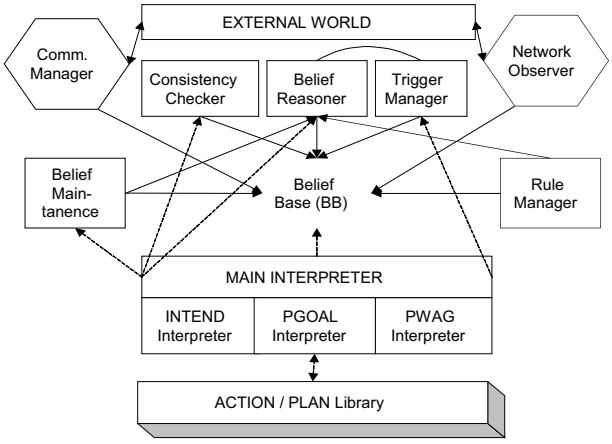

Figure 2: STAPLE Architecture

2.2 Communicative Acts Planner

The literature on the semantics of communicative acts [11, 12] based on JI theory defines two primitive communicative acts, REQUEST and INFORM. The goal of a *request* (REQUEST x y e *a* q t) is that the requestee y eventually does the action *a* and also comes to have a PWAG with respect to the requester x to do *a*. The requester's and requestee's PWAG's are relative to some higher-level goal q. For a request to be generated, the requester x has to believe that the requestee y can perform the requested action a. The goal of an *inform* (INFORM x y e p t) is that the listening agent y comes to believe that there is mutual belief between it and the informing agent x that the proposition *p* is true. For the communicative act *inform* to be performed, the informing agent x has a precondition that it believes the proposition *p* and x does not believe that the informed agent y believes *p*. Other communicative acts such as INFORM-REF, INFORM-IF, ASK_REF, ASK-IF, KNOW-REF, KNOW-IF, PROPOSE, AGREE, REFUSE are composed using the basic communicative acts *request* and *inform* [12]. For example, the communicative act ASK-REF is defined as a REQUEST to perform INFORM-REF to know the value of an unknown variable (See Table 2). The STAPLE definitions for these communicative acts can be found in [1].

3. GENERATING COMMUNICATIVE ACTS USING JI THEORY

In this section we discuss how STAPLE can generate communicative acts using JI theory to handle over-answering, teamwork and dynamic changes in the environment.

3.1 Planning relevant questions

A STAPLE agent for room reservation [5] would encode a joint intention between user and the agent for reserving a room. The steps of this plan could be to get all the required information and then process the knowledge base (KB) to get available rooms. In order to do this, the agent creates persistent goals to come to know the value of each required variables. Such variables include the desired room date, time, size, etc. Given that there is more than one such goal, this lack of knowledge will lead the agent to create concurrent goals to find the values. In order to do so, it will examine who it believes knows these values. For this example, we assume that the agent believes that the user knows the value for the date, time, size etc. The agent will then reason that it needs to obtain the information from the user in order to execute successfully the joint plan to reserve a room. There is no necessity for the programmer to explicitly tell the agent when to pose a question – it will determine for itself which information it has, and which it needs to obtain.

INTEND (agent, *REQUEST* (agent, user, action (*informref* (user, agent, i (D, date (Room,D)))
PGOAL (agent, done (action (*ask-ref* (agent, user, i(D, date (Room, D))*
PGOAL (agent, (*KNOW-REF* agent, i (D,date(Room,D))) *
PGOAL(agent, ∃ Rm (reserved(Rm,Date,Time,Size))
PWAG(agent, user, user (reserved(Rm, Date, Time,…)))) [plus predicates to provide variable typing information]

* The PGOAL is actually a bit more complex – the system wants to know the referent of the date that the user wants the room reservation.

Table 2: Plan Execution in a STACK

Concurrent stacks are established for the concurrent goals in the system. One such stack for obtaining the desired date is shown above (Table 2), there are similar stacks generated for time, size etc. The agent has a goal to know the value for the variables and also believes that user has the requisite information. The agent, by plan decomposition (as shown in Table 2) intends to send a *request for informref (wh-question)* to the user to know the value for the variable.

The joint goal (PWAG) entails an individual goal (PGOAL) to *know-ref* [6] the value for the required variable. The precondition for the plan *ask-ref* is satisfied if the agent believes that someone knows the value for the desired room reservation date (in fact, it thinks the user does). The plan *ask-ref* is a *request* for *informref* and finally the appropriate request is planned. Therefore there is no necessity for the programmer to specify precisely when to do a *request* or an *inform*, the agent automatically generates the requisite communicative acts as a consequence of its rational action [11, 12]. If a higher level goal succeeds (e.g., the system comes to know the date the user wants to reserve the room), the rest of the stack can be pruned. If the intention to do the action fails, the persistent goal still exists and the agent will try to find an alternate solution to know the value (if a solution exists) . For example, if there is some other agent or user who knows the value, the agent will plan a request to the other agent or user. STAPLE generates an disjunction of all possible ways to satisfy its goal and tries alternate actions if the first one fails.

If we are not careful, the presence of other concurrent goals could result in the generation of multiple potential requests to the user at the same time. STAPLE handles these parallel goals by a *controller stack*. Because STAPLE has only one verbal effector (its "mouth"), only one request can be sent at a time. The request can contain a single data field ("please say the date you want the room") or more ("please say the date and time you want the room").

3.2 Handling Over-Answering

In the present example, the controller stack initially contains all the goals that the agent come to know the various values it needs for the plan to be successfully executed, i.e., the desired date, time, room size etc. A prompt from the agent and a reply from the user could be:

Agent: What date would you like to book the room? *User: I would like a room for 50 people on March 28th.*

This is a typical example of over-answering (and commonly observed in user responses). The agent has received information for both the date and size. An *inform* from the user about the user's desires establishes mutual belief between the user and agent regarding the user's belief about the desired date. The agent now updates its knowledge base accordingly. The goals for obtaining the desired room date and size are discharged by the relevant triggers because they each have been achieved. These triggers remove the achieved goals from their respective stacks, as well as from the controller stack that depends on them. The system then determines the order of goals on its stack, leading to planning of subsequent speech acts and analysis of user responses. This goal-based methodology provides a general solution for handling over-answering because there is no necessity to code when and to whom to pose a request. Once all the required values are obtained, the agent tries to execute the reservation action.

3.2 Multi-agent interactions

Unlike the other approaches to dialogue discussed above, STAPLE can handle some aspects of multi-agent agent dialogue with the same mechanism. For example, the agent might have to obtain other information from a third agent in order to complete the room request. For example, it may need to ask a construction agent if ongoing renovations will be completed in time for the desired reservation date. STAPLE can handle this three-way interaction by directing requests (or wh-questions) to the appropriate agent or user.

3.3 Handling dynamic changes

JI theory helps in human-computer dialogue for tracking some aspects of dynamic changes in a multi-agent environment. For example, the agent and user might be at the point of agreeing to reserve an appropriate room. At that moment the agent realizes (from an external source or another agent's *inform* act) that the room is subject to renovation on the required date. This

makes the original joint commitment (of booking this particular room on that date) impossible. By the definition of joint intention, the agent creates a PGOAL to make the status mutually believed with the user. Through backward chaining, this PGOAL is achieved through an intention to *inform* the user of the goal's status. There is no necessity for us to explicitly program such cases since the general mechanism of tracking joint goals is built into the theory itself. The other methodologies need to keep track of such changes explicitly by deliberately programming the escape conditions.

4. RELATED AND FUTURE WORK

As far as collaborative dialogue is concerned, STAPLE is most related to ARITMIS [13] and Collagen [14]. Collagen is a combination of an intentional part (SharedPlans theory [8]) and an attentional part (focus stack) to keep track of the dialogue Each segment on the focus stack is associated with a SharedPlan. The segment contains the speech-acts or actions that the agent can perform. The main job of discourse interpretation in Collagen is to consider how the current direct communication or observed manipulation action can be viewed as contributing to the current discourse purpose, i.e., the purpose of the top segment on the focus stack. Collagen provides a generic framework for recording the decisions made and communicated by the agent (and the user), but not for making them. For example, the agent would not pose a request automatically even if it believes that the user knows the value for something the agent wants to know; the decision making is to be done explicitly.

The ARTIMIS system [13] was one of the early successes of integrating communication language with the intention model to achieve dialogue and inspired the present research. The system incorporates modal logic belief reasoning with a rational agent for speech-act understanding and generation in a spoken dialogue system. The present system can replicate the basic speech act reasoning that ARTIMIS performs, but supplies additional goals for collaborative problem solving and joint intention understanding.

TRIPS [15] is a system that tackles dialogue problems such as planning, mixed-initiative dialogue, indirect speech acts etc. It is one of the few systems that are capable of temporal reasoning. It contains a set of formally defined speech acts to handle collaborative problem solving. STAPLE is similar in principles to that of TRIPS, the differences are likely to come down to the practicalities of dialogue.

STAPLE is most related to STEAM [16] for teamwork capabilities. STEAM has explicit representation of team goals, plans and joint commitments, and also uses shared plan theories [8]. One difference between our work and STEAM is that STAPLE explicitly reasons about beliefs of other agents, and uses JI theory in generating communicative acts, whereas STEAM uses a fixed sequence of messages to create and discharge joint commitments. STEAM has proven to be very useful for building agent teams but the lack of belief reasoning and reasoning about semantics of communicative acts makes it difficult for STEAM to be used as a dialogue engine.

We have shown that a joint intention interpreter can extend existing dialogue research by modeling some aspects of human – human dialogue such as reasoning about who should be requested to supply information or to do an action. We also showed that by having a joint intention, the collaborating agent generates goals to establish mutual belief regarding the status of its commitments. There are other aspects such as implicit/explicit confirmations, clarification, negotiation etc. that we are currently trying to generalize.

5. REFERENCES

1. Kumar, S., "A Formal Semantics of Teamwork and Multi-agent Conversations as the Basis of a Language for Programming teams of Autonomous Agents", *[PhD Thesis]: Oregon Health and Sciences University, 2006.*
2. Kumar, S., Huber, M. J., Cohen, P. R., and McGee, D. R. "Toward A Formalism For Conversation Protocols Using Joint Intention Theory" *Computational Intelligence, 18(2):174-228., 2002b.*
3. Subramanian, R.A., Kumar, S., and Cohen, P.R., "Integrating Joint Intention Theory, Belief Reasoning, and Communicative Action for Generating Team-Oriented Dialogue", *To appear, AAAI 06, 2006.*
4. Seneff, S., Polifroni, J., "Dialogue Management in the Mercury Flight Reservation System," *presented at Satellite Dialogue Workshop, ANLP-NAACL, Seattle, April 2000.*
5. Bohus, D., and Rudnicky, A., "RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda", *Eurospech-2003, Switzerland.*
6. Rao, A. S. and Georgeff, M. P. "Modeling Rational Agents within a BDI Architecture", *In Proc. of 2nd Int. Conference on Knowledge Representation and Reasoning, 1991.*
7. Grosz, B. J. and Sidner, C. L. "Plans for discourse", *In P. R. Cohen, J. Morgan, and M. E. Pollack, Eds. Intentions in Communication", MIT Press, Cambridge, 417-444, 1990.*
8. Levesque, H. J., Cohen, P. R., and Nunes, J. H. T. "On Acting Together" *In Proceedings of AAAI-90, 94-99, 1990.*
9. Cohen, P. R. and Levesque, H. J. "Intention Is Choice with Commitment" *Artificial Intelligence, 42: 213-261, 1990.*
10. Cohen, P. R. and Levesque, H. J. "Teamwork*", Nous, 25(4): 487-512, 1991.*
11. Allen, J. F. and Perrault, C. R. "Analyzing Intention in Dialogues", *Artificial Intelligence, 15(3): 143-178, 1980.*
12. Cohen, P. R. and Perrault, C. R. "Elements of a Plan-Based Theory of Speech Acts", *Cognitive Science, 3(3): 177-212, 1979.*
13. Bretier, P. and Sadek, M. D. "A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction", *In J. P. Muller, M. J. Wooldridge, and N. R. Jennings, Eds. Intelligent agents III, LNAI, 1996.*
14. Rich, C., Sidner, C. L., and Lesh, N. B. "COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction", *AI Magazine. 22: 15-25, 2001.*
15. George Ferguson and James Allen, "TRIPS: An Intelligent Integrated Problem-Solving Assistant," *in* Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98*), 567-573, Madison, WI, 1998.*
16. Tambe, M. "Agent architectures for flexible, practical teamwork", *In Proceedings of AAAI-97, 22-28, 1997a.*