A Reactive-Deliberative Model of Dialogue Agency

David R. Traum

UMIACS, University of Maryland
A. V. Williams Building
College Park, MD 20742 USA

traum@cs.umd.edu

Abstract. For an agent to engage in substantive dialogues with other agents, there are several complexities which go beyond the scope of standard models of rational agency. In particular, an agent must reason about *social attitudes* that span more than one agent, as well as the dynamic and fallible process of plan execution. In this paper we sketch a theory of plan execution which allows the representation of failure and repair, extend the underlying agency model with social attitudes of mutual belief, obligation, and multi-agent plan execution, and describe an implemented dialogue agent which uses these notions, reacting to its environment and mental state, and deliberating and planning action only when more pressing concerns are absent.

1 Overview

For autonomous agents that operate in a realm of heterogeneous agents (including human agents), an agent theory should allow many of the features of natural language dialogue. The agent communication protocols should allow flexible turn-taking and initiative, reactions to the utterances of others, and flexible mechanisms of establishing mutual understanding and repairing problems when they occur. Such an agent must have (at least implicitly) functionally equivalent mechanisms to many of the mental and social attitudes of human agents.

This paper relates several strands of work towards developing a comprehensive agent theory. In section 2, the basic framework for deliberative agency is presented, including a theory of plan execution which makes explicit the mental attitudes that an agent has towards aspects of a plan that is being executed. The theory also allows the representation of different kinds of repair of plan executions as part of the reasoning and acting process. In section 3, this theory of agency is expanded to include *social agency*, in which an agent theory must also include inter-agent social attitudes. Three attitudes are discussed: mutual belief, specifically the *grounding* problem of adding to these mutual beliefs in dialogue, *obligations*, which are used to model the effects of certain speech acts and interactive behavior, and form another source for intentions, in addition to the individual goals of the agent, and *multi-agent plan executions*, the expansion of the theory in section 2, which relies on obligations, and is also used to model the grounding process. In section 4, an implemented social agent is briefly described: the dialogue manager of the TRAINS-93 system. These social attitudes are used in a reactive manner to guide the agent's behavior through the deliberative process in a local, step-by-step manner which is sensitive to the changing conditions as the dialogue progresses.

2 Aspects of Individual Agency

The approach to agency is informally sketched below.[1] A more full exposition in terms of a situation logic was presented in [24, 21]. The attitudes of **belief** and **desire** form the starting points for reasoning and acting. From the desires (and with reference to beliefs and other goals and intentions), the agent will *deliberate* and choose a set of **goals**: conditions that the agent will try to achieve. *Planning* or *means-ends reasoning* will lead to the formation or selection of a **plan recipe** designed to meet the goals. Recipes are viewed as consisting of a set of actions coupled with a set of constraints relating various properties of these actions (e.g. constraints on relative timing or objects and locations of actions, preconditions or effects represented as events or states which must hold over times related to the times of actions). We denote the set of actions of a recipe, R, by **Actions**(R). The set of constraints on a recipe will be denoted by **Constraints**(R)[2].

There are two kinds of "action" that must be considered in reasoning about plan execution: actual occurrences in the world (which we term here *executions*, noted with Greek letters), and abstractions of these occurrences (called *actions*) that are reasoned about by agents, and are the objects of intention, and components of recipes. For example, whenever an agent does something, there will be many abstractions of what was done, only some of which will be important for the plan. Agents will observe executions and decide whether or not particular actions of interest have been performed. What we call *actions* are sometimes called *action types* [12] or *activities* [3].

We use the symbol "$\triangleright$" to represent the "realizes" or "is characterized by" relation between an execution and an action. An action *occurs*, iff there is some execution which realizes that action: $\mathrm{Occurs(a)} \equiv \exists\alpha : \alpha \triangleright \mathrm{a}$

An execution could realize unrelated action types, and whenever an execution realizes an action type, it realizes a more abstract action type as well. As an example, suppose there is some execution α which realizes a MakeSpaghetti action: $\alpha \triangleright \mathrm{MakeSpaghetti(Agt, time)}$. It would then also be the case that this same execution realizes a MakeNoodles action: $\alpha \triangleright \mathrm{MakeNoodles(Agt, time)}$. It might also be the case that this same execution also realizes some unrelated action, say $\alpha \triangleright \mathrm{WakeUpDog(Agt, time)}$. In this case, we have two separate actions performed in the same execution.

Plans are viewed as individuals: platonic objects, which, like physical objects, can have their structure modified through time. At any given moment, a plan will correspond to a particular plan recipe. We denote the recipe of a plan, P, at time t as **Recipe**(P, t). Agents can be in particular relationships with plans, for instance, believing that they correspond to particular recipes, or intending to carry them out. When an agent *adopts* or *commits to* a plan, the agent will have a **commitment** to achieve (or maintain) the constraints of the plan and **intention** to perform the actions in the plan.

When an agent has committed to a plan, she can *try* to perform an action in that plan. She can also *observe* the situation she finds herself in, perhaps revising her beliefs and desires. She can also *replan*, revising the plans she is executing to correspond to different

[1] The approach presented here was based on the BDI model introduced in [5].

[2] Many of these constraints will be implicit in any *Representation* of a recipe, e.g. as shared variables in two separate actions, or domain constraints on possible recipes.

plan recipes, and thus changing her commitments and intentions. We distinguish two types of plan revision: *plan elaboration*, in which additional actions (e.g., decompositions of non-primitive actions) or constraints are added to a plan, and *plan repair*, in which some of the actions or constraints are removed. We can also distinguish the action of *plan repair* from *changing plans*. In both cases, the intentions and commitments of an agent change, but in the former case, the agent is changing the structure of a given plan which she continues to execute, whereas in the latter, the agent drops all intentions related to the old plan and adopts a new set.

We represent (future-directed) *intention* as having two parameters other than the agent: an action which the agent intends to perform, and a plan which this action is intended to (in part) achieve – $\mathbf{Intends}(\mathrm{Agt}, \mathrm{a}, \mathrm{P})$. This is roughly equivalent to the expression Int(Agt,By(a,P)) used by [19, 14]. *Commitment* is a weaker notion than intention. If an agent is committed to a proposition, this commitment will influence future deliberation to avoid actions which will result in the negation of the proposition holding, and if the agent comes to believe that the proposition will not hold, this will motivate deliberation and potential adoption of new intentions. Unlike intentions, however, commitments will not, in and of themselves, involve future action by the agent (although they may serve as the source for the adoption of intentions to perform actions, as well as being the metric by which the success or failure of such actions is judged). We represent commitment as a relation between an agent and a proposition, $\mathbf{Committed}(\mathrm{Agt}, \phi)$. The **committed** relation is similar to the *Int.Th* operator of [13], while **intends** is like their *Int.To*. Finally, we represent present-directed intention as an abstract action in its own right, called *try*. $\alpha \rhd \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P})$ means that the occurrence α is characterized by **Agt** attempting to do action **a** (intentionally) as part of executing Plan **P**.

From the point at which a plan is adopted to the point at which the intentions and commitments are dropped, we say that the agent is *executing* the plan. Plan execution is a generalization of Pollack's notion of an agent *having* a plan [19, 18], to the case in which the agent is in the midst of execution. This has important consequences for the component mental attitudes, since, e.g., an agent will no longer intend to perform an action it has already completed. In addition, we drop some of the *rationality conditions*, such as the following: *A believes that he can execute each act in* π *[The Plan].* and *A believes that executing the acts in* π *will entail the performance of* β *[The goal].* While violations of these conditions will be good reasons for the agent to repair the plan or execution, we do not believe that these should be constraints on the definitions of having and executing a plan. This allows us to model irrational agents, as well as agents following delegated plans, and agents in the midst of their repair process (e.g., after noticing that an actions would not be executable, but before repairing the execution).

A *Plan Execution Situation* (PES) is a situation in which a plan is being executed. It is a piece of the world containing a plan and the relevant agents, plan, objects, and events that make up executions or attempted executions of the plan. Each PES will have a designated current reference time, **now**. We represent a plan execution situation (named PE) as a 5-tuple, $[\mathrm{Agt}, \mathrm{P}_{\mathrm{PE}}, \mathrm{E}_{\mathrm{PE}}, \mathrm{Bind}_{\mathrm{PE}}, \mathrm{S}_{\mathrm{PE}}]$ where:

Agt represents the agent that is executing the plan.
P_{PE} represents the plan that this is the execution *of*.

E_{PE} represents the set of executions which have been performed in executing the plan. I.e., $\alpha \in E_{PE}$ iff $\exists a : \alpha \triangleright \mathbf{Try}(Agt, a, P_{PE})$.

$Bind_{PE}$ represents the *instantiation* (partial) function from actions of the plan to performed executions. For each action in the plan, its instantiation is an execution which the agent believes realizes that action. We will generally view this function as a set of relationships of the form: $a_i \rightsquigarrow \alpha_i$ where $a_i \in \mathbf{Actions}(\mathbf{Recipe}(P_{PE}, \mathbf{now_{PE}}))$, $\alpha_i \in E_{PE}$.[3] We will use a superscript version, $a_i \overset{PE}{\rightsquigarrow} \alpha_i$ as shorthand to indicate that $a_i \rightsquigarrow \alpha_i \in Bind_{PE}$.

S_{PE} represents the execution status, either 1 or 0, representing whether or not the plan is being executed.

Since each PES has its designated **now** time we will generally omit the temporal parameters from the recipe designators for the PES's plan. Thus **Recipe**(P_{PE}) will be used as shorthand for **Recipe**$(P_{PE}, \mathbf{now_{PE}})$. Similarly, we will use **Actions**(P_{PE}) to refer to the set of actions in the plan's recipe at the **now** time: **Actions**$(\mathbf{Recipe}(P_{PE}, \mathbf{now_{PE}}))$, and **Constraints**$(P_{PE})$ will stand for the set of constraints of the recipe at the **now** time: **Constraints**$(\mathbf{Recipe}(P_{PE}, \mathbf{now_{PE}}))$.

When an agent believes that an action in the plan has been performed by some execution (physical performance by the agent), we say that plan action is **instantiated**. Two predicates will be useful for classifying actions in a PES's plan as to whether they have been performed or not:

Definition 1 **Instantiated**(a, PE) *iff* $a \in$ **Actions**$(P_{PE}) \wedge \exists_{\alpha \in E_{PE}} a \overset{PE}{\rightsquigarrow} \alpha$

Definition 2 **Uninstantiated**(a, PE) *iff* $a \in$ **Actions**$(P_{PE}) \wedge \neg$**Instantiated**(a, PE)

Note that an action that is not in the plan at all is neither instantiated or uninstantiated with respect to the PES. An agent, **Agt**, is said to be executing plan **P** iff all of the following conditions hold:

1. Agt intends to perform all uninstantiated actions in plan P
2. Agt is committed to the occurrence of all constraints in P
3. Agt believes all instantiated actions have been realized by executions
4. Agt is committed to this realization of instantiated actions by executions

We define the remaining acts of a plan execution situation, **RActs**(PE) as those actions in the plan which have not been instantiated by any of the executions – these are the actions which still remain to be executed in order for the plan execution situation to be completed. Formally, **RActs**(PE) $\overset{def}{=} \{a_i \mid \mathbf{Uninstantiated}(a_i, PE)\}$

We can distinguish at least three distinct notions of the culmination of plan execution:

Action Completion – all actions in the plan have been performed: $\mathbf{RActs}(PE) = \{\}$

Successful Completion – all actions in the plan have been performed and all of the constraints have been met as well.

[3] Actually, it might take several executions to instantiate an abstract action, depending on how executions are divided up, so the range should actually be a subset of E_{PE} rather than a single element.

Goal Satisfaction – the goals which motivated adoption of the plan have been achieved.

Note that Successful Completion and Goal Satisfaction are somewhat independent. It may be the case that the goals are met before the plan is completed (e.g., imagine a plan to open an elevator door by pushing a button: the goal is to get the elevator door open, but suppose it opens by itself when someone else gets out). Depending on the plan adoption procedures, a plan might also be successfully completed without having satisfied the goals. A plan is an entity independent from the goals for which it was adopted, and the same plan could be used to try to achieve many different sets of goals. If a goal is not a constraint of the plan, it may be the case that the plan is successfully executed but the goals are not met. In a distributed control scenario, a plan executor may not have access to actual goals and may simply adopt particular plans under orders of a superior.

A plan executor will monitor the success of the plans it is executing, and engage in a repair if the plan is not successful. But the plan executor will also need to monitor goals: plan repair might also be warranted if the situation changes. This could include eliminating unnecessary actions if the goals are met though events external to the plan itself.

The *updating* of a situation by an occurrence is a situation that includes both the old situation as well as the occurrence. Generally this will involve adding an occurrence which happens at the **now** point of the situation, and the **now** point is then updated to immediately after the time of the occurrence. We introduce a situation updating function, $\mathbf{Update}(\alpha, \mathrm{PE})$ which takes as parameters an occurrence and a situation, and returns the updated situation. There are several ways in which an execution α may update a plan execution situation PE. The simplest is if the execution is intended to instantiate one or more of the actions in $\mathbf{RActs}(\mathrm{PE})$. We define a predicate **DoNext** over an execution and a PES which is true when there is a second PES which is an update of the first with the execution and binding relationship added:

Definition 3 $\mathbf{DoNext}(\alpha, \mathrm{PE})$ *iff*[4]

$$\begin{aligned}
&\exists \mathrm{a} \in \mathbf{RActs}(\mathrm{PE}) : \alpha \rhd \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P_{PE}}) \wedge \\
&\quad \exists \mathrm{PE}' : (\mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE})) \wedge (\mathbf{Recipe}(\mathrm{P_{PE'}}) = \mathbf{Recipe}(\mathrm{P_{PE}})) \wedge \\
&\qquad (\mathrm{E_{PE'}} = \mathrm{E_{PE}} \cup \{\alpha\}) \wedge (\mathrm{Bind_{PE'}} = \mathrm{Bind_{PE}} \cup \{\mathrm{a} \rightsquigarrow \alpha\})
\end{aligned}$$

We can define a *failure* as an execution which does not realize the action which it was intended to.

Definition 4 $\mathbf{Fail}(\alpha)$ *iff* $\exists \mathrm{Agt}, \mathrm{a}, \mathrm{P} : \alpha \rhd \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P}) \wedge \neg(\alpha \rhd \mathrm{a})$

What is more relevant for plan execution than a normative characterization of failure is *perceived failure*. This is an execution which the agent believes is to be failure: any executions of a PES which are not bound to an action in the plan.

Definition 5 $\mathbf{PerFail}(\alpha, \mathrm{PE})$ *iff* $\alpha \in \mathrm{E_{PE}} \wedge \neg\exists \mathrm{a_i} : (\mathrm{a_i} \in \mathbf{Actions}(\mathrm{P_{PE}}) \wedge \mathrm{a_i} \overset{\mathrm{PE}}{\rightsquigarrow} \alpha)$

[4] For simplicity, the definition presented here allows an execution to realize only a single action in the plan. The actual definition replaces the bound action, a, with a subset of $\mathbf{RActs}(\mathrm{PE})$.

If an agent believes that her action is a failure, the updated PES will include the execution, but will not have any new binding relationships.

There are at least two ways of continuing a plan execution other than simply updating the plan by performing a next action. One is to change the instantiation function such that either some action a_i in the plan which was previously instantiated by some some execution α is no longer instantiated by that execution, thus requiring a new execution to bind that action in any completed plan, or, conversely, that some a_i which was previously uninstantiated is newly bound by an execution α already part of the PES. Other instantiations in the plan execution would remain the same. We call this **ERepair**, standing for *execution repair*, since the same plan is maintained but the beliefs are changed about the success of previous action, and thus intentions about future actions are also changed.

Definition 6 $\textbf{ERepair}(\alpha, \mathrm{PE})$ *iff* $\exists \mathrm{PE}' : \mathrm{PE}' = \textbf{Update}(\alpha, \mathrm{PE}) \wedge$
$\textbf{Recipe}(\mathrm{P}_{\mathrm{PE}'}) = \textbf{Recipe}(\mathrm{P}_{\mathrm{PE}}) \wedge \mathrm{E}_{\mathrm{PE}'} = \mathrm{E}_{\mathrm{PE}} \wedge \mathrm{Bind}_{\mathrm{PE}'} \neq \mathrm{Bind}_{\mathrm{PE}}$

Another type of plan execution repair is to modify the plan itself, so that it is composed of a different recipe, though it still uses some of the same executions. This might also entail further **ERepair**, if it eliminates actions which have already been instantiated. We call this **PRepair**, standing for Plan Repair, since we are changing the plan that is being executed. In a **PRepair** the agent has changed intentions from performing the actions in the old recipe to performing actions in the new one.

Definition 7 $\textbf{PRepair}(\alpha, \mathrm{PE})$ *iff*
$\exists \mathrm{PE}' : \mathrm{PE}' = \textbf{Update}(\alpha, \mathrm{PE}) \wedge \textbf{Recipe}(\mathrm{P}_{\mathrm{PE}'}) \neq \textbf{Recipe}(\mathrm{P}_{\mathrm{PE}}) \wedge \mathrm{E}_{\mathrm{PE}'} = \mathrm{E}_{\mathrm{PE}}$

A plan execution may also terminate without success. An execution *cancels* a PES if it changes the execution status from one to zero. Formally:

Definition 8 $\textbf{CANCELS}(\alpha, \mathrm{PE})$ *iff*
$\exists \mathrm{PE}' : \mathrm{PE}' = \textbf{Update}(\alpha, \mathrm{PE}) \wedge \mathrm{S}_{\mathrm{PE}} = 1 \wedge \mathrm{S}_{\mathrm{PE}'} = 0$

Cancelling a PES amounts to the agent involved dropping the appropriate intentions to perform the plan by doing the specified actions. This doesn't necessarily indicate dropping the intention to achieve the goal behind the plan, and in fact the agents may start a new plan execution using the same recipe.

Given this basic framework, the activity itself is up to the agent: observing the success or failure of its actions, deciding to do a new step in the plan, repair an execution, or replan. Rationality principles such as those by by Pollack (presented above), or those used in [6] will be important guidelines for a successful agent. However, it is important to distinguish these principles of rationality which guide an agent's deliberation from the *definition* of the mental attitudes which constitute agency, in order to reason about violations of the rules and irrational agents. In a heterogeneous world, an agent may very well come across irrational agents, and, in fact, an agent may even be *obliged* by another agent to act in an irational manner (see section 3.2).

3 Social Agency

In order to model multi-agent interactions, such as natural language conversation, the model of agency described in the previous section must be extended in several directions. A social agent model must contain *social* attitudes in addition to the individual-level attitudes (e.g., belief, intention, etc). Social attitudes contain more than one agent as primary holder – they express a social relationship between the agents rather than just the cognitive state of one agent. In this section, we discuss three such social attitudes, and how they are important for reasoning about agent interaction such as dialogue.

3.1 Mutual Belief and Grounding

Mutual belief is one commonly assumed social attitude. While it is still somewhat controversial how best to formally model mutual belief (e.g., as a conjunction of nested beliefs or a self-referential attitude, see [4]), an interesting question is how such mutual belief gets established among interacting agents. While it has long been known that is impossible to guarantee the establishment of mutual knowledge in an environment where message transmission could fail [15], most formulations of speech acts have gone to the other extreme, and assumed mutual belief after the performance of any utterance within a shared situation (e.g., as the effect of a single agent speech act). However, examining actual conversation reveals that there is a process of feedback that accompanies initial utterances, and, in task-oriented spoken language, it is generally only after some sort of acknowledgment that an assumption of mutual belief is made. Furthermore, lack of understanding can be signaled with some sort of repair or request for repair. In cases in which the original speaker does not receive any feedback, one can observe requests for acknowledgment or repetitions and refashionings of the original contribution in an attempt to elicit some kind of feedback.

While the assumption of mutual belief resulting directly from a single utterance can be seen simply as an idealization (like modeling belief with a modal logic such as S5, with the resulting properties of logical omniscience), it is one with unfortunate consequences if used as the basis for models of speech acts and inter-agent communication. The consequences are twofold: first, a reasoner will make incorrect predictions about the mental state at particular times, and more importantly, the agent will be unable to recognize the relevance of or necessity for performing the feedback acts which actually establish the mutual understanding.

Clark and Shaefer call this process of reaching mutual belief (or common ground) *grounding* [8]. They present a descriptive model, in terms of presentation and acceptance phases that allow them to track the augmentation of common ground as the conversation proceeds. Their model is not well-suited for an on-line agent involved in dialogue, however, since it requires examination of subsequent spans of text in order to determine the boundaries of these phases.

In [22, 21], we developed a computational account of the grounding process. This account was based on speech act theory, using actions to introduce, acknowledge, and repair material. Traditional speech acts, such as *inform*, and *request* are now seen as multi-agent actions, which require participation by both parties to have their full effects (such as the mutual belief that one speaker wanted the other speaker to believe something).We introduced a level of dialogue structure called *discourse units* (DUs), at which these core speech acts are completed. These DUs are built up by single-utterance *grounding acts*. Recognizing the fact that multiple types of action occur in conversation, we extended speech act theory to the multi-level *conversation act* theory, described in [25]. As well as the grounding and core speech acts, there are also levels to model turn-taking behavior and higher order coherence of dialogue. A finite automaton was used to track the state of a DU, given a sequence of grounding acts in conversation. This model could also be used to predict possible subsequent acts as well as determine which act(s) must be performed in order to have a grounded DU (which would thus realize the effects of the constituent core speech acts).

3.2 Obligations

We claim that *Obligations* are necessary for modeling many social situations including natural language conversation, e.g., for capturing the effects of some speech acts, such as requests [23]. Obligations represent what an agent *should* do, according to some set of norms; its formal aspects are examined using Deontic Logic (e.g., [27, 17]). Generally, obligation is defined in terms of a modal operator often called *permissible*. An action is *obligatory* if it is not permissible not to do it. An action is *forbidden* if it is not permissible.

Just because an action is obligatory with respect to a set of rules R does not mean that the agent will perform the action. So we do not adopt the model suggested by [20] in which agents' behavior cannot violate the defined social laws. If an obligation is not satisfied, then this means that one of the rules must have been broken. We assume that agents generally plan their actions to violate as few rules as possible, and so obligated actions will usually occur. But when they directly conflict with the agent's personal goals, the agent may choose to violate them (and perhaps suffer the consequences of not meeting his obligations). Obligations are quite different from and cannot be reduced to intentions and goals. In particular, an agent may be obliged to do an action that is contrary to his goals (for example, consider a child who has to apologize for hitting her younger brother). [11] use obligations in a similar way, noting also that authority (such as a pre-existing hierarchical relationship) can be important in the ability to force obligations on others.

In [23] we argued that obligations play an important role in accounting for many of the interactions in dialog. For example, Table 1 shows the obligations resulting from the performance of speech acts. Obligations do not replace the plan-based model of speech acts (e.g., [10, 1]) but augment it. The resulting model more readily accounts for discourse behavior in adversarial situations and other situations where it is implausible that the agents adopt each others' goals. The obligations encode learned social norms, and guide each agent's behavior without the need for intention recognition or the use of shared plans at the discourse level. While such complex intention recognition may be required in some interactions, it is not needed to handle the typical interactions of everyday discourse.

The deliberation process in a social situation must take obligations into account, in addition to goals and intentions. In forming new intentions, sometimes an agent will choose to pursue his obligations rather than his goals. It is important for the agent to

source of obligation	**obliged action**
S_1 Accept or Promise A	S_1 achieve A
S_1 Request A	S_2 address Request: accept **or** reject A
S_1 YNQ whether P	S_2 Answer-if P
S_1 WHQ P(x)	S_2 Inform-ref x
utterance not understood or incorrect	repair utterance
S_1 Initiate DU	S_2 acknowledge DU
Request Repair of P	Repair P
Request Acknowledgement of P	acknowledge P

Table 1. Sample Obligation Rules

reason about both of these notions, so that this choice can be made explicit (either in the agent design or by the agent itself). There is also an illuminating relationship between this deliberation process and the notion of initiative in dialogue. Following the initiative of the other can be seen as an *obligation-driven* process, while leading the conversation will be *goal-driven*.

3.3 Multi-agent plan execution

Another common social attitude is that of *Joint intention* [9] or *shared plan* [14]. These concepts are used to model the propensity of a collaborative team to act. The intuition here is that it is more than just a collection of individual intentions and beliefs that is responsible for the coordinated teamwork activity. In [21], we developed a similar notion, that of agents executing a multi-agent plan. This is an extension of the notion of executing a plan described in section 2, above. A group of agents $\{A_i\}$ is executing a multi-agent plan **MP** iff:

1. Each A_i is executing a single-agent plan $\mathbf{MP}_i$, which has as its actions A_i's actions from **MP**, and as its constrains the constrains of **MP**, as well as the occurrence of all actions by other agents (thus A_i will be committed to the occurrence of the actions of others).
2. Each A_i is obliged to the other agents to perform her own actions as part of the multi-agent plan.

This formulation has several differences from the other works mentioned. First, no mutual belief is stipulated as a necessary component of the multi-agent plan execution. While mutual beliefs may sometimes be important for collaboration, and particularly for decisions about adopting plans and repairing plan executions, they are not strictly necessary for this kind of teamwork. In this framework, it is the personal commitment to the occurrence of the actions of others, and the obligations *to* those others (as well as the personal intention to perform the action) that forms the glue among the collaborating team. While any agent may break the team at any time by dropping these commitments and intentions, the obligations will remain until the agent is released, and it is this which motivates such actions as letting another know that an action has been performed or is deemed impossible. Even with notions of mutual belief of intentions (as in the SharedPlan formalism of [14, 13]), or commitment to inform an agent if an action is performed or impossible (as in the joint intentions of [9]), it is hard to see, in practice, what keeps an agent adhering to these commitments when its personal goals diverge.

This formulation was also used to formally model the grounding process described in section 3.1, above. Agents involved in a task oriented dialogue are assumed to be executing a specialization of the abstract plan recipe shown in Figure 1. This abstract recipe will be called CR (for Communication Recipe). Our claim is that successful execution of (a specialization of) this recipe will result in the (mutually assumed) mutual belief between the two agents that INITIATOR(CR) has communicated CONTENT(CR) to RESPONDER(CR). Agents engaged in conversation can be modeled as executing multi-agent plans which are specializations of this recipe. We will call any plan which has as its recipe a specialization of CR a *conversation plan*.

$$\begin{aligned}\textbf{Actions}(\mathrm{CR}) = \{&\mathrm{present_i} \mid \mathrm{present_i} = \mathrm{Present}(\mathrm{Agent1_i}, \mathrm{Content_i}, \mathrm{Recipient1_i}, \mathrm{t1_i})\} \cup \\ \{&\mathrm{ack_i} \mid \mathrm{ack_i} = \mathrm{Ack}(\mathrm{Agent2_i}, \mathrm{Content_i}, \mathrm{Recipient2_i}, \mathrm{t2_i})\}\end{aligned}$$

$$\begin{array}{ll} \textbf{Constraints}(\mathrm{CR}) = & \{ \\ \quad \text{Temporal constraints} & \forall \mathrm{i}(\mathrm{Before}(\mathrm{t1_i}, \mathrm{t2_i})), \\ \quad \text{Agent constraints} & \forall \mathrm{i}(\mathrm{Agent1_i} = \mathrm{Recipient2_i} = \mathrm{INITIATOR}(\mathrm{CR}), \\ & \quad \mathrm{Agent2_i} = \mathrm{Recipient1_i} = \mathrm{RESPONDER}(\mathrm{CR})), \\ \quad \text{Object Constraints} & \mathrm{CONTENT}(\mathrm{CR}) = \cup_{\mathrm{i}}\mathrm{Content_i}\} \end{array}$$

Fig. 1. Plan Recipe for Communication (Recipe CR)

The acts of presenting and acknowledging the content are broken into some indeterminate number of conceptual sub-acts, about at the granularity of the propositions in [16]. The only constraints on performance of these is that for a particular piece of the content, the presentation must come before the acknowledgment, and both must occur to achieve mutual belief. Constraints on exactly what types of executions can present and acknowledge the above contents will be determined by conventions of the particular language used and the communicative contexts. For any given execution, some parts of the content will be expressed explicitly as part of the compositional conventional meaning of the utterance, and others will be presented implicitly by conventions of situated meaning and Gricean implicatures. In [21], the grounding acts from [22] were given formal specifications as executions related to conversation plans.

4 An Example Implemented Social Agent

The dialogue manager of the TRAINS-93 system [2, 26] is implemented as a reactive deliberative agent. A rich model of the mental and conversational state (including private, nested, and mutual beliefs; private, proposed and shared plans; conversational goals, intentions, and obligations) is maintained and updated as the conversation progresses.

This includes adopting new beliefs and other attitudes as a result of the language interpretation process, as well as through the system's own reasoning and deliberation. The TRAINS-93 System is a large integrated natural language conversation and plan reasoning system. Its task is to develop and execute a shared plan about a transportation and manufacturing domain, through conversation with a human user. The dialogue manager module is responsible for maintaining the flow of conversation and making sure that the conversational goals are met. For this system, the main goal is that a shared plan which meets the user's domain goals is constructed and executed in the domain. The dialogue manager must track the state of the dialogue, determine effects of observed conversation acts, generate utterances, and send commands to the domain task reasoner when appropriate.

In designing an agent to control the behavior of the dialogue manager, we choose a *reactive* approach, in which the agent is constantly making local decisions as to what to do next, rather than planning whole interactions in advance. So-called "discourse plans", specifying the sequence of utterances to be performed may be appropriate in a text generation domain, but in dialogue one can not completely predict the responses of the other agents involved. Moreover, timely behavior is critical: the same response can have a very different connotation if it is delayed. Still, deliberation over the range of individual and social attitudes will be an important component in the agent's activity.

The TRAINS system is very cooperative and generally lets the user take the initiative whenever the user would like to. Consequently, obligations are made a higher priority than working on the system's own goals. When deciding what to do next, the agent first considers obligations and decides how to update the intentional structure (add new goals or intentions) based on these obligations. Obligations might also lead directly to immediate action. If there are no obligations, then the agent will consider its intentions and perform any actions which it can to satisfy these intentions. If there are no performable intentions, then the system will deliberate on its overall goals and perhaps adopt some new intentions (which can then be performed on the next iteration).

For the dialogue agent, special consideration must be given to the extra constraints that participation in a conversation imposes. This includes some weak general obligations (such as acknowledging utterances by others and not interrupting) as well as some extra goals coming from the domain setting to maintain a shared view of the world and the domain plans which are to be executed. We prioritize the sources for the deliberations of the actor as follows:

1. Discourse Obligations (e.g., answer a question, repair a misinterpretation)
2. Weak Obligation: Don't interrupt user's turn
3. Intended Speech Acts
4. Weak Obligation: Grounding (coordinate mutual beliefs)
5. Discourse Goals: Domain Plan Negotiation
6. High-level Discourse Goals

The implemented agent serializes consideration of these sources, looking at the lower priority items only if there is nothing requiring attention in the higher priorities. Moreover, only one action is performed at a time before re-checking the context. The updating of the conversational context due to perceived conversation acts or actions of other modules of the system progresses asynchronously with the operation of the discourse agent.

Whenever the dialogue agent is active, it will first decide on which task to attempt, according to the priorities given above, and then work on that task. After completing a particular task, it will again search for the most urgent task, although by then the context may have changed due to, e.g., the observation of a new utterance from the user. The agent is always running and decides at each iteration whether to speak or not (according to turn-taking conventions); the system does not need to wait until a user utterance is observed to invoke the agent, and need not respond to user utterances one by one.

The agent's first priority is fulfilling obligations. If there are any, then the agent will do what it thinks best to meet those obligations. If there is an obligation to address a request, the agent will evaluate whether the request is reasonable, and if so, accept it, otherwise reject it, or, if it does not have sufficient information to decide, attempt to clarify the parameters. In any case, part of meeting the obligation will be to form an intention to tell the user of the decision (e.g., the acceptance, rejection, or clarification). When this intention is acted upon and the utterance produced, the obligation will be discharged. Other obligation types are to repair an uninterpretable utterance or one in which the presuppositions are violated, or to answer a question. In question answering, the agent will query its beliefs and will answer depending on the result, which might be that the system does not know the answer.

In most cases, the agent will merely form the intention to produce the appropriate utterance, waiting for a chance, according to turn-taking conventions to actually generate the utterance. In certain cases, though, such as a repair, the system will actually try to take control of the turn and produce an utterance immediately. For motivations other than obligations, the system adopts a fairly "relaxed" conversational style; it does not try to take the turn until given it by the user unless the user pauses long enough that the conversation starts to lag. When the system does not have the turn (priority 2), the conversational state will still be updated, but the agent will not try to deliberate or act.

When the system does have the turn, the agent first (after checking obligations) examines its intended conversation acts (priority 3). If there are any, it calls the NL generator to produce an utterance.[5] It might not be convenient to generate all the intended acts in one utterance, in which case some intended acts may be left for the future consideration.

If there are no intended conversation acts, the next thing the agent considers is the grounding situation (priority 4). The agent will try to make it mutually believed (or *grounded*) whether particular speech acts have been performed. This will involve acknowledging or repairing user utterances, as well as repairing and requesting acknowledgment of the system's own utterances. Generally, grounding is considered less urgent than acting based on communicative intentions, although some grounding acts will be performed on the basis of obligations which arise while interpreting prior utterances.

If all accessible utterances are grounded, the agent then considers the negotiation of domain beliefs and intentions about the TRAINS world, (priority 5). The agent will try to work towards a shared domain plan, adding intentions to perform the appropriate speech acts, including accepting, rejecting, or requesting retraction of user proposals,

[5] If the only intention is to acknowledge, the agent will postpone the generation until it checks whether there is any other content, such as an acceptance or answer, that could be expressed in the same utterance.

requesting acceptance of or retracting system proposals, and initiating new system proposals or counterproposals. The agent will first look for User proposals which are not shared. If any of these are found, it will add an intention to accept the proposal, unless the proposal is deficient in some way (e.g., it will not help towards the goal or the system has already come up with a better alternative). In this latter case, the system will reject the user's proposal and present or argue for its own proposal. Next, the agent will look to see if any of its own proposals have not been accepted, requesting the user to accept them if they have been simply acknowledged, or retracting or reformulating them if they have already been rejected. Finally, the agent will check its private plans for any parts of the plan which have not yet been proposed. If it finds any here, it will adopt an intention to make a suggestion to the user.

If none of the more local conversational structure constraints described above require attention, then the agent will concern itself with its actual high-level goals (priority 6). For the TRAINS system, this will include making calls to the domain plan reasoner and domain executor, which will often return material to update the system's private view of the plan and initiate its own new proposals. It is also at this point that the agent will take *control* of the conversation, pursuing its own objectives rather than responding to those of the user.

Finally, if the system has no unmet goals that it can work towards achieving, it will hand the turn back to the user or try to end the conversation if it believes the user's goals have been met as well.

4.1 Example

The following example gives a sense of how the dialogue manager uses this representation of context and priorities to engage in dialogue. More extended examples are presented in [23, 21]. The example starts with a declarative utterance by the User:

U: "There are oranges at Corning."

At the core speech act level, this is interpreted as initiating both an **inform** (about the location of oranges), and a **suggestion** that the oranges be used in the current plan. At the grounding level, this is seen as the initiation of a DU. It is also seen as keeping the turn. This has the following effects on the context - first (at priority level (4), there is an unacknowledged DU, which will require grounding. More prominently, however, the user still has the turn, so the system will just wait for the next utterance.

U: "Is a boxcar there?"

This is interpreted as asking a yes-no question, continuing the current DU, and releasing the turn. Now there is an additional core speech act in the ungrounded DU, and the system has the turn. The chosen action is now, at priority 4, to add the intention to acknowledge the content in this DU (a new item at priority 3). Forming this intention also causes the system to update its mental state with the effects of this content. In this case, the inform and suggestion will be interpreted as user proposals, at priority level 5. The YNQ leads to an obligation to answer the question, which is at priority level 1. Since the obligation is of highest priority, the system acts upon this by querying its beliefs to see if a boxcar is at Corning. This check returns negatively, which leads the system to intend to inform the user of this fact. Now, the the highest priority are the intended speech acts. These are passed to the NL generator, and a combined, acknowledgment/answer is provided with:

S: “No there isn’t”

This simple example displays some of the flexibility of the reactive agency model. Given different responses or a different initial mental state, many variants of this simple dialogue could have been produced using the same rules. Most of the flexibility of plan-based approaches is maintained, while the obligation model presents a much more direct account of question answering, without any need for reasoning about or adopting the desires or intentions of the user.

5 Summary

This paper has briefly sketched a theory of agency suitable for an agent in a multi-agent domain. For flexibility, an agent’s communication language should include most (if not all) of the properties of natural language. An agent must be able to reason about the execution process, including repairing and revising plans, when necessary. Section 2 described the basics of a theory of plan execution. Social, as well as individual, level aspects of mental state, as discussed in section 3, are important to reason about multi-agent interaction. Both reactive and deliberative aspects are important for an agent in an uncertain world. Reactivity is crucial to deal with changing circumstances in a timely manner, but deliberation is also important to be able to do complex tasks. These two components can be integrated, as in the TRAINS system, by having some of the “reactions” be to deliberate on a particular problem. This deliberation must also be constrained to more local decisions rather than solving a whole problem, in order to allow the system to react in a timely manner.

References

1. James F. Allen and C. Raymond Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.
2. James. F. Allen, L. K. Schubert, G. Ferguson, P. Heeman, C. H. Hwang, T. Kato, M. Light, N. Martin, B. Miller, M. Poesio, and D. R. Traum. The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:7–48, 1995.
3. Cecile T. Balkanski. Modelling act-type relations in collaborative activity. Technical Report 23-90, Harvard University Center for Research in Computing Technology, 1990.
4. Jon Barwise. *The Situation in Logic*, chapter 9: On the Model Theory of Common Knowledge. CSLI Lecture Notes: Number 17. Center for The Study of Language and Information, 1989.
5. Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.
6. P. Bretier and M. D. Sadek. 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, 1996. In this volume.
7. Herbert H. Clark. *Arenas of Language Use*. University of Chicago Press, 1992.

8. Herbert H. Clark and Edward F. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259–294, 1989. Also appears as Chapter 5 in [7].
9. Phillip R. Cohen and Hector J. Levesque. Confirmations and joint action. In *Proceedings IJCAI-91*, pages 951–957, 1991.
10. Phillip R. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.
11. F. Dignum and B. van Linder. Modeling social agents: Communication as action. 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, 1996. In this volume.
12. Alvin I. Goldman. *A Theory of Human Action*. Prentice Hall Inc., 1970.
13. Barbara [J.] Grosz and Sarit Kraus. Collaborative plans for group activities. In *Proceedings IJCAI-93*, pages 367–373, 1993.
14. Barbara J. Grosz and Candace L. Sidner. Plans for discourse. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.
15. J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.
16. Jerry Hobbs. Ontological promiscuity. In *Proceedings ACL-85*, pages 61–69, 1985.
17. L. Thorne McCarty. Permissions and obligations: An informal introduction. Technical Report LRP-TR-19, Dept. of Computer Science, Rutgers University, 1986.
18. Martha E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.
19. Martha E. Pollack. *Inferring Domain Plans in Question-Answering*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 1986.
20. Yoav Shoham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings AAAI-92*, pages 276–281, 1992.
21. David R. Traum. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, Department of Computer Science, University of Rochester, 1994. Also available as TR 545, Department of Computer Science, University of Rochester.
22. David R. Traum and James F. Allen. A speech acts approach to grounding in conversation. In *Proceedings 2nd International Conference on Spoken Language Processing (ICSLP-92)*, pages 137–40, October 1992.
23. David R. Traum and James F. Allen. Discourse obligations in dialogue processing. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, 1994.
24. David R. Traum and James F. Allen. Towards a formal theory of repair in plan execution and plan recognition. In *Proceedings of the 13th Workshop of the UK Planning and Scheduling Special Interest Group*, September 1994.
25. David R. Traum and Elizabeth A. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599, 1992. Special Issue on Non-literal language.
26. David R. Traum, L. K. Schubert, M. Poesio, N. G. Martin, M. Light, C. H. Hwang, P. Heeman, G. Ferguson, and J. F. Allen. Knowledge representation in the TRAINS-93 conversation system. *International Journal of Expert Systems*, to appear 1996.
27. G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

This article was processed using the LaTeX macro package with LLNCS style