Models of Plans to Support Communication: An Initial Report

Appears in Proceedings of AAAI-90

Models of Plans to Support Communication: An Initial Report*

Karen E. Lochbaum
Barbara J. Grosz
Harvard University
Cambridge, MA 02138
kel@pandora.harvard.edu

Candace L. Sidner
DEC Cambridge Research Lab
Cambridge, MA 02139
sidner@crl.dec.com

Abstract

Agents collaborating to achieve a goal bring to their joint activity different beliefs about ways in which to achieve the goal and the actions necessary for doing so. Thus, a model of collaboration must provide a way of representing and distinguishing among agents' beliefs and of stating the ways in which the intentions of different agents contribute to achieving their goal. Furthermore, in collaborative activity, collaboration occurs in the planning process itself. Thus, rather than modelling plan recognition, per se, what must be modelled is the *augmentation* of beliefs about the actions of multiple agents and their intentions. In this paper, we modify and expand the SharedPlan model of collaborative behavior (Grosz and Sidner, 1990). We present an algorithm for updating an agent's beliefs about a partial SharedPlan and describe an initial implementation of this algorithm in the domain of network management.

Introduction

Agents collaborating to achieve a goal bring to their joint activity different beliefs about ways in which to achieve the goal and the actions necessary for doing so. Each agent may have incomplete or incorrect beliefs. In addition, their beliefs about each other's beliefs and capabilities to act may be incorrect. As a result, collaborative activity cannot be modelled by simply combining the "plans"[1] of individual agents. Even when the agents' beliefs are the same, a simple combination is not possible (Grosz and Sidner, 1990). An adequate model of collaboration must provide a way of representing and distinguishing among agents' beliefs and of stating the ways in which the intentions of different agents contribute to achieving their goal.

In this paper, we modify and expand the SharedPlan model of collaborative behavior originally proposed by two of the authors (Grosz and Sidner, 1990), present an algorithm for updating an agent's beliefs about a partial SharedPlan, and describe an initial implementation of this algorithm in the domain of network management. This work is being done in the context of a project to investigate the development of systems that support coordination of graphical and linguistic means of communication between agents involved in collaborative activities [see also (Reiter, 1990; Marks, 1990; Marks and Reiter, 1990; Balkanski *et al.*, 1990)]. This paper sets forth an initial framework for modelling particular aspects of collaborative activity and identifies several key problems.

The sample dialogue in Figure 1 illustrates collaboration in the network management domain. In this discourse, the network manager (NM) and the network presenter (NP) are working together to determine what type of maintenance to perform on a particular switching node that can no longer handle the amount of traffic flowing through it. NM begins by stating the problem and then proceeds to ask for information that would be useful in solving it. NP supplies that information, both verbally and graphically, while also making further suggestions. A goal of our work is to provide the basis for a computer system to assume the role of NP.

(1) NM: It looks like we need to do some maintenance on node39.
(2) What kind of switch is it?
(3) NP: It's an XYZ, but it's at full capacity.
(4) NM: OK, then let's replace it with an XYZ+.
(5) First, we'll have to divert the traffic to another node.
(6) Which nodes could be used?
(7) NP: *[puts up diagram]*
(8) Node41 looks like it could temporarily handle the extra load.
(9) NM: I agree.
(10) Let's go ahead and divert the traffic to node41 and do the replacement.

Figure 1: Sample discourse

Most previous models of plan recognition (Allen and Perrault, 1980; Kautz, 1990; Sidner, 1985) are inadequate for modelling collaboration because they make assumptions appropriate for single agent plans, or the

*This research has been supported by a contract from U S WEST Advanced Technologies and by a Bellcore Graduate Fellowship for Karen Lochbaum.

[1] We have scare quotes around "plan" because it has been given a variety of meanings in the AI literature.

plans of multiple independent agents, but not for the plans of collaborating agents. These assumptions include a data-structure model of plans as well as a master/slave relationship between all-knowing agents. In particular, these models treat plans as sequences of steps to be performed; as each step occurs, it is filled into a particular plan schema. These models also assume that one agent, the speaker, is a controlling agent, while the other agent, the hearer, is simply a reactive agent, inferring and cooperating with the plan of the speaker. Because the hearer and speaker are assumed to have the same complete and correct knowledge of the domain, the systems do not distinguish between the speaker's and hearer's beliefs about actions.

Pollack, in modelling how an inferring agent reasons about another agent's invalid plans, differentiates each agent's individual beliefs and intentions regarding actions and the relations among them from other agents' (Pollack, 1986; Pollack, 1990). Her model thus provides a useful base on which to define a model of collaborative activity. However, she also makes the assumptions that the inferring agent has complete and accurate knowledge of domain actions and that that agent is recognizing the plans of another. In collaborative activity, collaboration occurs in the planning process itself. Thus, rather than modelling plan recognition, per se, what must be modelled is the *augmentation* of beliefs about the actions of multiple agents and their intentions.

In the next section, we present the modified definition of SharedPlan and describe two new constructs: recipes and the Contributes relation. We then present an augmentation algorithm and give examples of its use. Finally, we describe future directions of this work.

Definition of SharedPlan

The definition of SharedPlan follows Pollack's work on single agent plans (Pollack, 1986; Pollack, 1990) in taking the notion of "having a plan" to be central and to consist of being in a certain mental state, namely, holding certain beliefs and intentions regarding acts and their executability. This stance differs from other work in planning which takes a plan to be a data structure encoding a sequence of actions. The mental state view is crucial to the ability to model plans constructed and carried out collaboratively.

SharedPlans are defined in terms of *act-types* and relations among them. We distinguish types of actions, act-types (or *acts*), from individual tokens of the type (for which we will use the term *action*). An act-type is a triple, $\langle\gamma(\mathrm{P}_1 \ldots \mathrm{P}_n)$,G,T$\rangle$, where the P_i are parameters of the activity $\gamma(\mathrm{P}_1 \ldots \mathrm{P}_n)$[2], G is the agent who performs the activity, and T the time interval over which the activity occurs (Balkanski *et al.*, 1990)[3]. Act-type relations include generation [CGEN (Pollack, 1986; Pollack, 1990)] and enablement. In addition, complex act-types, for example, ones involving sequences of acts or simultaneous acts, may be built from simpler ones using act-type constructor functions (Balkanski *et al.*, 1990).

Two agents, G_1 and G_2, are said to have a SharedPlan during time T1 to accomplish an action of type **A** during time T2 if and only if they hold the beliefs and intentions listed below:

SharedPlan(G_1,G_2,A,T1,T2) $\Longleftrightarrow$

1. MB(G_1,G_2,EXEC($\langle\alpha_j$,G_{α_j},$\mathrm{T}_{\alpha_j}\rangle$),T1)
2. MB(G_1,G_2,R:recipe-for-**A**,T1)
3. MB(G_1,G_2,INT(G_{α_j},$\langle\alpha_j$,G_{α_j},$\mathrm{T}_{\alpha_j}\rangle$,T1),T1)
4. MB(G_1,G_2,INT(G_{α_j},$\langle\alpha_j$,G_{α_j},$\mathrm{T}_{\alpha_j}\rangle\wedge$ Contributes($\langle\alpha_j$, G_{α_j}, $\mathrm{T}_{\alpha_j}\rangle$, **A**), T1), T1)
5. INT(G_{α_j},$\langle\alpha_j$,G_{α_j},$\mathrm{T}_{\alpha_j}\rangle$,T1)
6. INT(G_{α_j}, $\langle\alpha_j$,G_{α_j},$\mathrm{T}_{\alpha_j}\rangle\wedge$ Contributes($\langle\alpha_j$, G_{α_j}, $\mathrm{T}_{\alpha_j}\rangle$, **A**), T1)

In this definition, the index j ranges over the act-types in the recipe R for doing **A**. For each α_j, G_{α_j} denotes the agent who performs the activity, and T_{α_j} denotes the time interval over which the activity is performed. Each T_{α_j} is a subinterval of T2 which is the interval over which **A** is performed. The predicate MB holds of two agents, a proposition, and a time just in case the two agents mutually believe the proposition over the time interval. The predicate INT holds of an agent, an act-type, and a time if the agent intends to perform an action of that type during the time interval. EXEC holds if the agent is able to perform an action of the appropriate type (Pollack, 1990).

This definition provides a framework in which to further evaluate and explore the roles that particular beliefs and intentions play in collaborative activity. Currently, the definition only provides for recipes in which each constituent act is performed by one of two agents. However, we are currently investigating act-types performed by multiple agents (i.e. G_{α_j} is a set of agents) and extensions of the definitions of INT and EXEC to multiple agents.

The two most complex portions of the definition of SharedPlans are the recipe-for-**A** in Clause (2) and the Contributes relation in Clauses (4) and (6). Recipes were not part of the original SharedPlan definition (Grosz and Sidner, 1990) and Contributes was only informally defined; both of these are discussed in more detail below.

SharedPlans are constructed incrementally. When agents G_1 and G_2 have some partial set of beliefs and intentions from the SharedPlan definition (or even simply have a mutual desire to achieve a SharedPlan), but have not yet completed the building of such a

[2] When the parameters of an activity are not at issue, we will simply use the activity name, γ, as an abbreviation for $\gamma(\mathrm{P}_1 \ldots \mathrm{P}_n)$.

[3] We follow the Prolog convention of specifying variables using initial uppercase letters and constants using initial lowercase letters.

plan, they are considered to have a partial SharedPlan, which we will denote by *SharedPlan**. Each utterance of a discourse contributes some information about beliefs and intentions to this evolving jointly-held plan. As opposed to a SharedPlan*, a SharedPlan for a particular **A** specifies all of the beliefs and intentions necessary for performing **A**. It is not necessary, however, for the SharedPlan to be fully specified before any actions may take place. On the contrary, performance of actions may be interleaved with discussion of those and other actions.

Recipes

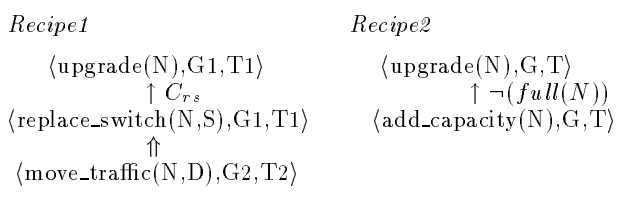
Following Pollack (Pollack, 1990), we use the term *recipe* to refer to what agents know when they know a way of doing something. Clause (2) of the SharedPlan definition thus states that when agents have a SharedPlan for doing some act, they must hold mutual beliefs about the way in which to perform that act. Recipes are specified at a particular level of detail. Hence, Clause (2) only requires the agents to have mutual beliefs about acts specified at the particular level of detail of the recipe and does not require them to have mutual beliefs about all levels of acts that each agent will perform. In our earlier work (Grosz and Sidner, 1990), we considered only simple recipes; each recipe consisted of only a single act-type relation. However, for any interesting activity, recipes include many different types of relations.

Recipes, more formally stated, are aggregations of act-types and relations among them. Act-types, rather than actions, are the main elements in recipes because the regularities about which agents can have beliefs must necessarily be stated in terms of types and not tokens. A recipe for performing a particular act, Γ[4], encodes constraints on constituent acts and their interrelationships. Performing all of the constituent acts in the recipe, following any ordering stipulated by the act-type relations, will result in the performance of Γ.

Figure 2 contains two sample recipes from the network management domain. Recipe1 indicates that one way to upgrade a particular switching node, subject to conditions, C_{rs}, is to replace the node by a new node of a different type; i.e. *CGEN(⟨replace_switch(N:node, S:switch_type),G1,T1⟩, ⟨upgrade(N:node),G1,T1⟩, C_{rs})*[5]. However, before that can be done, the switch traffic must be diverted to another node. According to Recipe2, a node may be upgraded by adding more capacity to it; however, there is a generation-enabling condition on this relation which requires that the node still have room for more capacity.

[4]When the specific components of an act-type are not at issue, we will abbreviate the triple by using capital Greek letters.

[5]We assume a typed logic in which *X:type* is used to indicate the *type* of object that may replace the variable *X*. For presentation purposes, the types have been omitted from Figure 2.



↑c indicates generation subject to the condition c
⇑ indicates enablement

Figure 2: Two network management recipes

Note that the act-types in these recipes are specified at a fairly high level of detail. For example, although the activity *move_traffic* is further decomposable into activities involving flipping specific switches, such low-level details are not part of this recipe. They may, of course, be part of a recipe for *move_traffic* itself. For example, this more detailed level is necessary for a situation in which a new operator is being taught exactly how to move traffic around a network.

Recipe1 may be used to upgrade any node, but Recipe2 may only be used for types of switches that allow some additional capacity to be added. This condition on applicability of recipes may be modelled by associating recipes with elements of an act-type lattice (Balkanski *et al.*, 1990). In such a lattice, act-types are partially ordered according to specialization — more specialized act-types lie below their abstractions. For example, such a lattice might contain the act-types *⟨upgrade(N:node),G,T⟩*, *⟨upgrade(N:node_type1),G,T⟩*, and *⟨upgrade(N:node_type2),G,T⟩*, where the second and third act-types are specializations of the first. The parameter type specialization of *node* to *node_type1* or *node_type2* corresponds to the distinction made by the recipes in Figure 2; switches of *node_type2* allow the addition of extra capacity while switches of *node_type1* do not. Thus, Recipe1 is associated with the lattice element *⟨upgrade(N:node_type1),G,T⟩*, while both Recipe1 and Recipe2 are associated with *⟨upgrade(N:node_type2),G,T⟩*. The distinction made in the use of these two recipes is not modeled in the recipes themselves, but is a consequence of the association of recipes with lattice elements.

Upon completion of a SharedPlan, the recipe-for-**A** in Clause (2) of the definition will be of the same form as the recipes in Figure 2. During the construction of a SharedPlan, however, the agents are establishing mutual beliefs about act-types relevant to performing **A**. Thus, at any time during this construction, the recipe-for-**A** of Clause (2) is only partially specified.

Contributes

Agents' beliefs about recipes may be partial in a number of different ways. Not only might an agent not know all of the act-types involved in performing **A**, but he might not know the exact relations that hold between the act-types. For example, Jack may believe that to replace his oil filter, he has to find

the drain plug on his oil pan (perhaps because his friend Carol, who believes that it's good practice to always change your oil and filter together, told him so), but he may not know the specific act-type relations that hold between finding the plug and replacing the filter (e.g. *enable(⟨find(P:drain_plug),G,T1⟩, ⟨remove(P:drain_plug),G,T2⟩)*,...). The Contributes relation used in Clauses (4) and (6) of the SharedPlan definition is a general act-type relation intended to capture this level of knowledge. Thus, Jack's beliefs about replacing his filter, would include an act-type relation of the form, *Contributes(⟨find(P:drain_plug),G,T1⟩, ⟨replace(F:oil_filter),G,T3⟩)*.

Contributes is defined as the transitive closure of the D-Contributes relation where D-Contributes depends upon the act-type relations and constructors defined in (Balkanski *et al.*, 1990) and is defined as follows:[6]

D-Contributes (Γ, Δ) ≡

1. $\rho(\Gamma, \Delta)$
   where ρ is one of the primitive act-type relations: generate,enable, facilitate, ...

OR

2. $\Delta = \kappa(\Gamma_1, \Gamma_2, \ldots, \Gamma_n)$, such that $\Gamma = \Gamma_j$ for some j, $1 \leq j \leq n$, and
   κ is one of the act-type constructor functions: sequence, simult, conjoined, or iteration.

That is, D-Contributes holds between act-types Γ and Δ, when Γ stands directly in an act-type relation ρ to Δ or when Γ is a direct component of Δ.

The Contributes relation is used in Clauses (4) and (6) of the SharedPlan definition as a modifier indicating the way in which an act-type is performed. That is, *INT*$(G_{\alpha_j}, \langle \alpha_j, G_{\alpha_j}, T_{\alpha_j} \rangle \wedge$ *Contributes*$(\langle \alpha_j, G_{\alpha_j}, T_{\alpha_j} \rangle$, **A**$)$, *T1)* means that G_{α_j} intends to do the activity α_j *as a way of* contributing to **A** (cf. Pollack's act-type constructor *by* (Pollack, 1986; Pollack, 1990)). Such a construction is meant to capture the notion that α_j is only being done as a way to achieve A, and hence failures related to it constrain replanning (Bratman, 1990).

SharedPlan Augmentation

The process of augmenting a SharedPlan* comprises the adoption of mutual beliefs and intentions related to the clauses of the SharedPlan definition. Such beliefs include those about acts in the recipe of the plan, properties of those acts, and intentions to perform them. A SharedPlan* may thus be affected by utterances containing a variety of information. An individual utterance, however, can only convey information about the beliefs or intentions of the speaker of that utterance. Thus, the algorithm for updating a SharedPlan* includes mechanisms for attributing individual beliefs and intentions and subsequently establishing mutual beliefs based on those individual attitudes and on the discourse and SharedPlan contexts in which the utterance occurs.

The basic algorithm for updating a partial SharedPlan on the basis of information contained in an utterance of a dialogue carried out in support of collaborative activity is given below. G_i and G_j denote the two agents, G_1 and G_2. G_i denotes the speaker of the utterance and G_j the other participant. The algorithm given is for G_j; G_i's differs in some details, e.g. G_i already knows his beliefs about recipes whereas G_j must infer new ones to attribute to G_i. $\lambda Act.Prop$ is used to denote the proposition (*Prop*) expressed in the current utterance, where *Act* indicates the particular act to which it refers. To simplify the discussion, we will ignore the details of the information about *Act* that *Prop* represents. Utterances (4) through (6) of the dialogue in Figure 1 illustrate the variety of such propositions that utterances may contain.

SharedPlan Augmentation Algorithm:
Assume:
Act is an action of type Γ,
G_i designates the agent who communicates $\lambda Act.Prop$,
G_j designates the other agent,
SharedPlan*(G_1, G_2, **A**, T1, T2).

1. As a result of the communication, assert MB(G_1,G_2,BEL(G_i,$\lambda Act.Prop$)).
2. Search own beliefs for BEL(G_j,$\lambda Act.Prop$).
3. Ascribe BEL(G_i,Contributes(Γ,**A**)).
4. Search own beliefs for Contributes(Γ,**A**) and where possible, more specific information as to how Γ contributes to **A**.
5. If Steps (2) and (4) are successful, signal assent and MB(G_1,G_2,Contributes(Γ,**A**))[7].
6. If Step(2) or Step (4) is unsuccessful then query G_i or communicate dissent.

Each agent brings to the joint activity private beliefs and intentions. As these private attitudes are communicated, they participate in the agents' having a SharedPlan. Steps (3) and (4) reflect this behavior. Namely, G_j, based upon G_i's utterance, *ascribes* beliefs to G_i and then searches his own beliefs with respect to G_i's. Taking "having a plan" to mean having a set of beliefs and intentions about an act is crucial to these steps. The interleaving of the ascription of private plans and of SharedPlans would be much more difficult if plans were taken to be data structures.

The Contributes relation also plays an important role in the ascription process in that it supports partiality during the construction of plans. Without this general relation, plan ascription would have to assert that some much stronger relationship held, one which

[6] We assume a theory of mutual belief that allows for belief of a Contributes relationship without necessitating explicit beliefs about all of the supporting D-Contributes relationships.

[7] G_j may, of course, suggest alternatives.

would not necessarily be supportable (e.g. if information relevant to establishing it were unavailable). Furthermore, when two agents have different recipes for achieving a goal, and are not initially aware of these differences, use of the Contributes relation is critical (see the second example below). If the inferring agent has no beliefs about how an act-type contributes to the goal, he can choose to accept the belief of a Contributes relation on good faith or after checking with the other agent. In the case that the inferring agent has beliefs that suggest that there is no relation between the act-type and the goal, he can dissent to the other agent's beliefs about that act-type.

Typically, only partial information is available to G_j when he is reasoning about G_i's utterances. Thus, an agent can only tentatively ascribe beliefs to other agents and not logically deduce their being held. Upon subsequent information, the agent must be able to retract his beliefs. We are currently investigating the use of defeasible reasoning and direct argumentation for the ascription of belief (Konolige, 1988; Konolige and Pollack, 1989).

An Example

To illustrate the algorithm, consider its use with respect to the dialogue of Figure 1 with NP as the reasoner in the augmentation algorithm. Using conversational default rules (Grosz and Sidner, 1990), from utterance (1) and NP's lack of dissent, the agents may infer *SharedPlan*(nm,np,upgrade(node39))*[8].

Given the context of the SharedPlan, NP must decide how utterance (2), which he could interpret as *Desire(nm,find_out(switch_type(node39)))*, relates to the goal of the SharedPlan, performing the act *upgrade(node39)*. Based on NP's knowledge of act-type specializations, particularly those described above (his utterances indicate that he does indeed have such knowledge), NP recognizes that NM may be asking this question to ascertain which recipes for upgrading a node are applicable. Thus, in his response in utterance (3), NP reports the type of the node. In addition, however, because he believes that NM will recognize that nodes of that type do allow the addition of extra capacity, he adds the caveat that the node is already at full capacity (Reiter, 1990). That is, although this type of node usually allows the addition of capacity, the condition that there be available capacity left cannot be met.

Assuming that NM also has similar knowledge about act-type specializations and recipes, one could interpret her "OK" as indicating recognition of the caveat and hence that Recipe2 is not applicable. Because the act-type, $\langle$*upgrade(N:node_type2)*$, G, T\rangle$ in the lattice has both Recipe1 and Recipe2 associated with it, however, NM may now proceed using her beliefs about the act-types and their interrelations given in Recipe1[9].

NP could interpret the remainder of utterance (4) as *Desire(nm,replace_switch(node39,xyz+))*. According to the SharedPlan augmentation algorithm, as a result of this communication, NP ascribes *BEL(nm,Contributes(replace_switch(node39, xyz+),upgrade(node39)))*. At this point, NP must search his own beliefs to determine if he also believes such a contributing relation. That is, NP considers the recipes he knows for *upgrade(node39)* (given the previous discourse, only Recipe1 is relevant at this point) and finds specifically that indeed he believes *CGEN(replace_switch(node39,xyz+), upgrade(node39),* C_{rs}*)*. NP can then indicate his belief of this contributing relation by either signaling assent or simply not signaling dissent. Given his lack of dissent, NM can assume that NP believes the action to contribute to the upgrade and thus *MB(nm,np,Contributes(replace_switch(node39, xyz+), upgrade(node39)))* is established.

From NM's next utterance, (5), NP could infer *Desire(nm,move_traffic(node39,D:node))*. To summarize the algorithm's performance, NP will search his own beliefs (i.e. Recipe1) and find that he believes *Contributes(move_traffic(node39, D:node), upgrade(node39))*. This Contributes relation is based upon the enabling relation between *move_traffic(node39,D:node)* and *replace_switch(node39,xyz+)*, which, in turn, was previously found to contribute to *upgrade(node39)*.

Utterances (6) through (9) comprise a sub-dialogue to find an appropriate node to which the traffic may be diverted (Litman and Allen, 1990; Sidner, 1985; Grosz and Sidner, 1986). Finally, with utterance (10) the complete recipe-for-**A** is spelled out.

A More Complicated Example

Unlike the previous example, we now assume that the two agents have different know-how. NM knows Recipe1, but NP knows only Recipe3, which is similar to Recipe1, but does not contain an act for replacing the switch. That is, according to Recipe3, a node may be upgraded by simply moving traffic off of it. We will use the modified dialogue in Figure 3 to illustrate the algorithm's performance in such cases.

From utterance (2), NP infers *Desire(nm,move_traffic(node39,D:node))*. Upon searching Recipe3, NP finds that he believes *CGEN(move_traffic(node39, D:node), upgrade(node39),* C_{mt}*)* and hence *Contributes(move_traffic(node39, D:node), upgrade(node39))*. The specific relationship he believes to hold between these two acts is different from what NM believes; however, this difference has not yet surfaced. After iden-

[8] Throughout this discussion, we will use an abbreviated notation in which only the activity of the act-type is specified; i.e., *SharedPlan*(nm,np,upgrade(node39))* is a shorthand for *SharedPlan*(nm,np,*$\langle$*upgrade(node39),*$\{nm,np\}$*, T2*$\rangle$*, T1, T2)*.

[9] The current augmentation algorithm only models that portion of the example which follows this point; however, we are investigating extensions which will model the preceding discussion as well.

tifying node41 as a possible D, NM continues, in utterance (6), with her recipe and indicates a desire to replace node39. Upon searching his beliefs, NP cannot determine how *replace_switch(node39,xyz+)* contributes to *upgrade(node39)*. He signals his confusion and asks NM to clarify why such an action is necessary.

(1) NM: It looks like we need to do some maintenance on node39.
(2) What node could we divert its traffic to?
(3) NP: *[puts up diagram]*
(4) Node41 looks like it could temporarily handle the extra load.
(5) NM: I agree.
(6) Let's divert the traffic to node41 and replace node39 with an XYZ+ switch.
(7) NP: Huh? Why do you want to replace node39?

Figure 3: Sample discourse

Implementation

An initial version of the augmentation algorithm has been implemented in Prolog in which the system is one of the agents working on a SharedPlan. In the context of a SharedPlan for some act **A**, when presented with a specification of the other agent's (i.e. the user's) desire of some act, Γ, the system searches its recipes for **A** to determine if a Contributes relationship holds between Γ and **A**. In this initial implementation, we have concentrated on the process of searching through recipes to determine whether or not an act-type is a constituent part of a recipe. Unlike previous work (Kautz, 1990; Pollack, 1990), however, the recipes through which the system searches involve more complex relations than simple step decomposition or generation. In addition, because the system may have multiple recipes for **A**, if no contributing relation can be found between Γ and **A** in a particular recipe, then that recipe is removed from consideration. If the user's recipes differ from the system's, then Γ may not be a constituent part of any of the system's recipes, in which case, the system will respond with "HuhΓ". The current system is able to model those portions of the above network management examples in which NP ascribes beliefs (both individual and mutual) based on NM's desire for a particular act-type.

Conclusion and Future Directions

In this paper, we have presented a model of collaboration and discussed an algorithm for augmenting an evolving jointly-held plan. We are currently investigating the following extensions: (1) The use of defeasible reasoning and direct argumentation for the ascription of belief (Konolige, 1988; Konolige and Pollack, 1989); (2) Modelling the agents' abilities to negotiate to an agreed upon act rather than simply assenting or dissenting to each other's suggestions; (3) Extending the formalism to include acts performed by more than one agent.

Acknowledgments

We would like to thank Cecile Balkanski, Joyce Friedman, Martha Pollack, and Stuart Shieber for many helpful discussions and comments on this paper.

References

Allen, J. and Perrault, C. 1980. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178.

Balkanski, C. T., Grosz, B., and Sidner, C. L. February 1990. Act-type relations in collaborative activity. Technical report, Harvard University.

Bratman, M. E. 1990. What is intention. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication*. MIT Press. Forthcoming.

Grosz, B. and Sidner, C. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3).

Grosz, B. and Sidner, C. 1990. Plans for discourse. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication*. MIT Press. Forthcoming.

Kautz, H. 1990. A circumscriptive theory of plan recognition. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication*. MIT Press. Forthcoming.

Konolige, K. and Pollack, M. 1989. Ascribing plans to agents: Preliminary report. In *Proceedings of IJCAI-11*, Detroit, Michigan.

Konolige, K. 1988. A direct argumentation system. In *Proceedings of the International Symposium on Machine Intelligence and Systems*, Torino, Italy.

Litman, D. and Allen, J. 1990. Discourse processing and commonsense plans. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication*. MIT Press. Forthcoming.

Marks, J. and Reiter, E. 1990. Avoiding unwanted conversational implicatures in text and graphics. In *Proceedings of AAAI-90*, Boston, MA.

Marks, J. January 1990. Automating the design of network diagrams. Technical report, Harvard University.

Pollack, M. E. June 1986. A model of plan inference that distinguishes between the beliefs of actors and observers. In *Proceedings of the 24th Annual Meeting of the ACL*.

Pollack, M. E. 1990. Plans as complex mental attitudes. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication*. MIT Press. Forthcoming.

Reiter, E. 1990. Generating action descriptions that exploit a user's domain knowledge. In Dale, R., Mellish, C., and Zock, M., editors, *Natural-language Generation (working title)*. Forthcoming.

Sidner, C. February 1985. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1):1–10.