A Planner for Plan Construction Dialogues for Multi-Agent Plans

Bryan McEleney, Gregory O'Hare

Department of Computer Science, University College Dublin, Dublin 4, Ireland

bryan.j.mceleney@ucd.ie, gregory.o'hare@ucd.ie

Abstract. A natural language dialogue planner is described that chooses dialogue moves to revise the beliefs of an agent. In particular those beliefs that refer to the mental state of another agent are revised. In a planning problem of repeated decisions, the future decision of the other agent is better predicted, and therefore the immediate plan decision of the first agent is a better one.

1 Introduction

The purpose of language is to get something done that wouldn't have occurred had it not been used. It involves two agents, one a speaker, whose unclear intention must be revealed to a hearer. If the hearer is cooperative, he might then perform some further action that together with the speaker, coherently and jointly achieves the intention. Recognition of intention can only come about through an observable physical or spoken action. Observed physical actions might contribute to some parent plan. If the observer believes that the plan cannot be completed by the agent, he will add a relevant contribution. In this way, the actor has used language, since he communicated his intention, but not in the proper sense, rather as a side effect of his execution of physical actions. Alternatively, with a spoken action, something similar happens, except that the act used by the agent has no physical purpose, rather it is a speech act [12]. Nevertheless it can be treated in a planning or plan recognition system exactly as a physical act would be [1].

Planning for a single agent has its roots in the STRIPS planning system [6], which has pervaded work on planning in dialogue as well as for physical problems. In multi-agent planning, a cooperative plan recognising agent will observe the common environment, opportunistically waiting for the other agent to commit an act that communicates his intention. This agent adds a contribution to the first agent's plan, and executes it. This happens recursively to produce a series of synchronised, alternating acts, realised as a turn-by-turn dialogue or a turn-by-turn sequence of physical moves, and not uncommonly a mixture of both. The process is illustrated in figure 1.

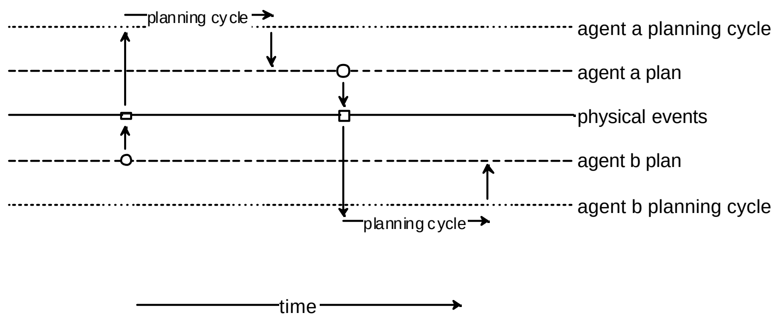

Fig. 1. Synchronisation of Planning, Execution, and Plan Recognition

To produce a multi-agent plan, a time-tree is a useful representation of the sequence of alternating choices made by each agent [11]. Similar to a decision tree, each node represents either a choice made by an agent at a particular time, or an uncertainty about his belief state or decision process. For example, in planning to ask a question, an agent could not be sure which answer will be given, but would like to choose a move that responds to that answer. This uncertainty is an important factor, since it requires a tree model of the *expected* multi-agent plan rather than a deterministic, linear model.

As a simple physical example, imagine a shopper, laden down with supermarket bags, who appears at the door of the supermarket. An observer, wishing to be cooperative, develops a time-tree plan that has an uncertainty point representing the belief that the shopper is capable of opening the door by himself. There is also a choice point representing the observer's choice to help by opening the door. There are four leaves in the time-tree – two of which are happy outcomes and two of which are not so happy. A much more complex example is a collaborative planning task, where, say, an architect and an engineer must pool their respective expert skills to build a multi-agent plan for constructing a house. This kind of plan is full of uncertainties, particularly about the plan rules that the other agent will apply in putting forward a coherent contribution to the plan. In both the simple and the complex examples, the agents might decide to use dialogue to eliminate the uncertainty nodes in the time-tree. In the simple example, the observer asks the shopper if he needs a hand, and in doing so the two not-so-happy outcomes are prevented from occurring. In the complex example, a much longer dialogue takes place, that compares the different candidate plans, eliminating many uncertainties as the dialogue progresses, remaining coherent between dialogue segments as the focus of the dialogue moves over the candidate plans. The dialogue must also be limited in length since there can be too many plan candidates, each with many uncertainty nodes, and diminishing returns in their resolution.

In this paper, a planner for such dialogues is described that can interact with a human dialogue partner, and make decisions about dialogue moves that maximise their value in the domain-level plan decision of the agent. It is a mixed initiative system in that both agents have identical roles, contribute actions symmetrically to the domain plan, and contribute dialogue moves symmetrically to the dialogue plan that constructs it. To be useful as a natural language dialogue system, there are equally important human cognitive constraints and linguistic requirements. The planner has two main sections – one that develops the domain-level plan, and one that, using the domain level plan as its subject, develops discourse-level plans [3].

2 Related Work

The earliest work on plan construction dialogue systems is that of Ramshaw [10]. Ramshaw's planner has a limited domain state knowledge, which is expanded by consultation of an expert, as the plan space is searched. Similar to here, the plan decomposition choices of the agent are represented as a tree. Control of the search is by an orderly tree traversal over the decomposition tree, with the agent using various queries to check precondition satisfaction, or to instantiate variables in the plan. Since there is no measure of plan utility, the agent backtracks when it fails to produce a correct domain-level plan, halting as soon as a correct one is found. As such there can be no parallel consideration of competing plans. In contrast to here, this is a system-initiative system, for constructing a single agent plan, which concentrates on the agent's domain state beliefs, stopping short of exploring plan rule beliefs.

Another development is the TRAINS project [2], in which a computer fulfils the role not of the planner, but of the domain expert, providing answers to queries about a planning problem involving a transportation network, and suggesting plan alternatives when it is appropriate. It is intended to be a demonstrable speech-based system in which both agents communicate in natural language. As part of the project, a corpus of human-human dialogues was collected. While the planning problem is not identical, the dialogue moves that appear in the corpus have been useful to verify the moves chosen for the planner presented here.

3 The Decision Mechanism

The domain-level planner takes a decision-theoretic view of the domain-level plans. Each achieved intention provides a quantitative reward or a cost to the agent. The uncertainty that one agent holds of another's beliefs is represented as a probability distribution. Planning then proceeds as the recursive selection, at every choice node in the time tree, of the branch that maximises expected utility. Since a probabilistic model of belief is used, a probabilistic plan recognition system based on belief networks [5] produces probabilistic intention hypotheses. For the dialogue plan, deciding whether to resolve an uncertainty is based on a well known rule for deciding among alternatives where complete information can be given about uncertain outcomes [8]. Here, a *critical point* is defined as a point of uncertainty in the belief model of the agent, whose resolution is expected to improve the agent's expected utility through changing his decision. Critical points occur in two places. First, the agent's own beliefs may be uncertain, and through passing information, a consensus about a belief can be obtained. Second, the uncertainty may be in the agent's model of the other agent's beliefs, on which the expectation of the other's action is based. To decide about revising an uncertainty, a Bayesian Updating rule is used to estimate a consensus value for a belief by assuming that each agent's estimate is conditionally independent of the true value of the proposition, and that each is independent of the other. These assumptions rarely hold in practice, yet they are necessary so that a figure can be obtained from the available statistics, and have been shown to be appropriate in, for example, Naïve Bayes Classifiers. Variables A1 and A2 represent each agent's probability estimates for A. With the assumptions and using Bayes' rule, the following formula is obtained:

$$P(A \mid A1, A2) = \frac{P(A \mid A1) . P(A \mid A2)}{P(A \mid A1) . P(A \mid A2) + P(\neg A \mid A1) . P(\neg A \mid A2)} \tag{1}$$

Consider again the domain planning model, and the time tree constructed by the agent. Each uncertainty node in the tree is a joint distribution over beliefs, represented compactly as a belief network. According to a dependent decision in the following choice node, the uncertainty node branches upon one of these beliefs. Each branch has a different utility value, but the expected utility value is the weighted sum over the outcomes, according to the "maximise expected utility" decision rule. The probability for the uncertain proposition is denoted by p. Now suppose that the agent is considering a number of plans, and that the one with the maximum expected utility is called *max*. Let one of the other plans be called *can*. Suppose *can* has two branches at the uncertainty node, one of high utility and one of low utility, say *can-favourable* and *can-unfavourable*. Suppose that:

utility(can) < utility(max)

but

utility(can-favourable) > utility(max)

Figure 2 shows the expected utility comparison of *can* and *max* as a function of p. Notice if p is known to be low, the agent should choose max, gaining and losing nothing, but if it is high, the agent should choose can since the expected utility surpasses that of max. That is, there is never a loss, but sometimes a utility gain obtained from knowing p. This is represented by the shaded area between the utility curve for *can* and the utility curve for *max*, from the value for p at which *can* and *max* have equal utility.

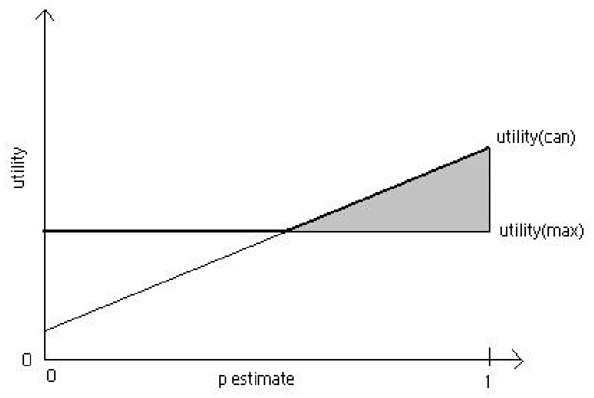

Fig. 2. Utility of *can* and *max*

The agent must estimate the utility gain from obtaining the other agent's estimate of p. It is assumed that his p estimate is normally distributed around the agent's own estimate. Then, the expected utility gain is obtained by taking the expected utility gain over this normal distribution, using formula 1 to determine p, by integrating. Where the gain is greater than zero, a critical point has been found, since there is something to be gained from resolving the uncertainty.

4 The Dialogue Planner

A number of basic dialogue moves have been identified which correspond with those used in natural planning dialogues, and which exp loit critical points. Each move has a simple internal structure whose primitive elements are dialogue acts named **inform**, **question**, **propose** and **pass-turn**, corresponding directly to single speech acts. Where a move operates on a belief, there are several variations - plain beliefs about the domain state, beliefs about the agent's preexisting intentions, and beliefs about plan rules covering preconditions and effects of actions, and enablement, threats, and decomp osition relations between actions. Queries may take yn- or wh- forms. Some moves cover one turn and some cover two.

The first move, *query-clarify*, constructs a direct query about the value of a critical point belief, which is answered by the other agent, using a **question** and an **inform**. The utility of the move is given directly by the utility gain of the critical point

The second move, *information-lend*, is a complement to query-clarify. It is used by the second agent to highlight a critical point that is known to the second but not known to the first. Such disparity happens because the agents have different belief sets and therefore different time-trees. Without information-lend, critical points could be overlooked by the first agent and never queried. It consists of just one dialogue act, an **inform**. Similar to query-clarify, the utility of the move is given immediately by the value of the critical point.

The third move, *propose*, declares a preference for a plan, and has several purposes. First, it obtains a conclusion to the dialogue if the other agent accepts. Second, it marks the focus of the dialogue around which subsequent moves can be arranged. Third, if the other agent does not accept, it prompts the other agent to use the fourth move, incongruity-attack. The utility gain is calculated as the indirect gain expected in a following incongruity-attack.

The fourth move is *incongruity-attack*. It is used where one agent believes that the other holds a misconceived preference for one plan over another. It can be prompted directly by a propose, or indirectly by abductive inference of the subject domain plan that motivated a previous information-lend or query-clarify, since a precondition to these two moves is that the subject plan is highly valued. The attack consists of a sequence of query-clarify and information-lend moves that treat the misconceived preference as something similar to a critical point. The utility gain is based on the criticality of this point, but is divided among the sequence elements that together seek out the misconception.

Finally, a simple move, *pass*, is used when the agent cannot come up with a move that offers a reasonable utility gain. The agent passes control to the other agent instead. If neither can think of a reasonable move, a propose is used by default, concluding the dialogue if agreement is obtained.

5 Move-level Control

At any point in the dialogue, there can be a number of applicable moves. One way to decide between them is to greedily choose the one that provides the maximum expected utility gain. A problem with this approach is that the utility of only the leading plans is boosted, increasing their suitability for selection in the next iteration. Other plans that are initially poor but potentially valuable are therefore not covered. Instead of the greedy strategy, softmax selection is used, which explores many plans in early iterations, but narrows consideration to the leading ones in later iterations [13].

A constraint that needs to be used in move selection is that the move is in focus [7]. The focus shift is defined as the number of hops on the time-tree between the focused choice node of the last move and the choice node of the current one. Common knowledge of the focus point is important so that moves are recognised and correctly interpreted by the hearer. Ensuring the hearer knows the focus point is costly, since a description of the subject plan contains a part for each choice node that determines it. With more hops, more choices must be described to ensure the hearer is at the new focus point. The importance of focus is confirmed in the TRAINS corpus [2], where speakers rarely switch to a different point of focus unless the first point has been exhausted, and where many moves cannot be used successfully without a plan context. For these reasons, a *focus-description-cost* is subtracted from moves that shift the focus.

6 Computer-Human Interaction

While the dialogue moves might be efficient according to the formal planning model, there is no guarantee that they will be understood by the human dialogue partner, and as some of these moves are for changing the partner's domain-plan decision, this can be a problem. The human partner is limited in understanding complex sets of arguments, in bringing to mind all of the branches of the time-tree and in remembering several plans or several arguments that together motivate a domain-level decision. Some techniques to ensure that moves will be convincing are to keep arguments short enough that the human can make a decision without forgetting some of the argument or finding the argument too complex. The computer should make moves that are not strictly critical, but highlight points the human would not have thought of by himself. In particular, propose helps to ensure coverage of the time-tree. Focus shifts should be more strictly avoided for memory restrictions as well as linguistic reasons, and when focus returns to a point, recapping might be used to restore the context. Again because of incomplete covering of the time-tree, a human might be more inclined to adopt a plan that was highlighted in a dialogue than one that was not mentioned, and so the computer should be careful that all alternatives are fairly brought to attention, whether they are thought to be the best ones or not.

7 Implementation and Evaluation Plan

Implementation of the planner in PROLOG is underway, with the domain-level time-tree planner available for use, and the dialogue planner to follow. The domain-level planner can choose between plans with uncertain outcomes, and is a useful tool in its own right since it can decide between ambiguous and unambiguous utterances or actions. For example, it can decide whether a risky utterance that uses an uncertain ellipsis will produce a more efficient dialogue than one that uses a safer, but longer utterance. The second component, the dialogue planner, takes as its input the time-trees used by the domain-level planner. Since the emphasis of the system is on deciding dialogue moves, and because it is a laborious and error-prone, uncontrolled natural language input is not possible for the dialogue planner. Input must be presented as a formal description of the dialogue act type, belief type, and the object actions or states of the belief, written in predicate-argument form. However, natural language generation is much simpler since canned text with slots can be used for each of the move specifications.

For evaluating the system, a test problem has been constructed. Two agents, who share a kitchen, must decide how to cooperatively cook a meal. One is an expert in desserts and the other is an expert in main courses. This problem is ideal since each agent has incomplete knowledge of the other agent's beliefs about the domain state and the plan rules they will use for their part of the task. Since a kitchen has numerous shared resources like ingredients and implements, each individual's plan interacts in many places with the other's, and so critical points will be numerous. The plan recognising domain planner will resolve some of these before they happen, for example, inferring that the use of an egg by the other agent implies the cooking of an omelette implying that the frying pan will be unavailable in the following turn. Each agent's belief state will be instantiated with a set of domain knowledge. Recursively nested models will be instantiated to model the beliefs agents have about one another's beliefs. At each level, a belief network is given that represents the probability of each belief state. Reasonable differences will be introduced between the beliefs of an agent, and the model of those beliefs held by the other, so that a noticeable utility gain is obtained by the use of dialogue moves. Each agent will have differing beliefs about the domain state. A number of problem instances with variations in the mental state of the agents will be used to give a representative set of samples that might occur in a real collaborative planning problem.

The evaluation measure for the system takes account of the value of the dialogue, using two terms – the quality of the plan produced, which is given directly by the utility value of the final time-tree, and the length of the dialogue. Dialogue length is important since a dialogue can be far too long if it is required to cover a large area of the agent's beliefs. An appropriate weighting is given to each term.

Evaluation of the system will be in two different rounds. The first will take advantage of the symmetry of the planning problem, by running the system against itself to produce dialogue sequences. This is an inexpensive way to find an initial tuning for the system parameters with an emphasis on efficiency, but is not ideal since it excludes the particular traits of human interaction. In the second round, the system will hold a dialogue with a set of human participants. This round serves an analytical purpose in that each time the human moves, a computer generated move is computed and compared. Any systematic oversight of a useful move type should be apparent in this analysis. This might even work in reverse, highlighting human deficiencies, particularly if a computer-computer dialogue for the same problem instance is more efficient. The second round is also evaluative, in that it directly measures the human-computer performance on the test set. The subjects will bring their own cooking knowledge to the experiment since it would be difficult to ask a subject to temporarily change their ingrained beliefs about cookery, in contrast to the computer which can be programmed with different belief configuration for each run.

8 Future Work

The planner presented here is useful for solitary problem instances, but where there will be several instances whose requisite domain knowledge intersects, and where dialogue is relatively less costly, the agent might choose moves that are not strictly critical for the given plan, since these moves can broaden the agent's knowledge for further planning problem instances. An extension to the planner has been proposed for computing the added value for these beliefs. The approach is straightforward because it uses the existing domain-level planner. Since beliefs about domain states and plan rules are probabilistic, a space of expected future plans can be generated using the planner. For each plan in the space, critical points are found, and the expected utility gain is calculated for each belief that can be resolved in the plan. Summed over the plan space, an expected utility value is then obtained for each belief in the agent's belief set, with which the belief is annotated. If the plan space is large, a Monte Carlo sampling approach can be used. This process should be repeated occasionally since the agent's beliefs change over time. Then, in calculating a critical point for a given plan instance, the stored annotations can be used to boost the utility of each point.

Another development has to do with belief argumentation. Currently, to revise a belief, two moves may be used – query-clarify and information-lend. Each of these moves is simple, since only the belief in question is attacked. However, beliefs are rarely independent, being connected to others in the belief network, which act as supporting evidence. Many negotiation dialogues go beyond the simple model to include argumentation about the evidence as well. While it is desirable, the current system will not provide a mechanism for argumentation since an existing system [9, 4] could be slotted in place of the two moves that are currently used. However, argumentation dialogues can be long and unpredictable – it is important to decide whether an agent should embark on one, or just revise the root belief without question. Two sources of utility need to be considered in this decision. The first source represents the general knowledge gained in revising the evidence beliefs, and this can be computed using the annotation method explained above. The second source represents the greater confidence in the value of the root belief since its evidence is bolstered. One approach to this is to model the estimation error of each evidence random variable. The error is propagated from children to parents in the belief network. Error causes "spill" of probability mass effectively reducing the size of the shaded area where a utility gain is obtained (figure 2). As each variable is revised, its error diminishes, reducing the spill. A neighbourhood of supporting beliefs could be calculated for which the spill reduction outweighs the cost of the dialogue, with argumentation limited to the neighbourhood.

9 Conclusion

A natural language dialogue planner for plan construction dialogues was described. Using the time-tree output from a probabilistic domain-level planner, the planner makes quantitative decisions about the dialogue moves that best improve the domain planning decision. The design is motivated by direct utility gains obtained in a formal planning model, linguistic requirements in the planning of efficient dialogues, and human requirements in the understanding and generation of complex dialogue moves.

References

1. Allen J., Perrault, R. : Analyzing Intention in Utterances. Artificial Intelligence 15 (1997) 148-178

2. Allen, J, Schubert, L., Ferguson, G., Heeman, P., Hwang, C., Kato, T., Light, M., Martin, N., Miller, B., Poesio, M., Traum, D. : The TRAINS Project: A Case Study in Building a Conversational Planning Agent. Journal of Experimental and Theoretical AI, 7 (1995) 7-48

3. Carberry, S., Lambert, L., A Tripartite Plan-Based Model of Dialogue. Proceedings of the 29th Annual Meeting of the ACL (1991) 47-54

4. Chu-Carroll, J., Carberry, S., Collaborative Response Generation in Planning Dialogues. Computational Linguistics 24(3) (1998) 355-400

5. Charniak, E., Goldman, R. : A Bayesian Model of Plan Recognition. Artificial Intelligence 64 (1) (1993) 53-79

6. Fikes, R., Nilsson, N., STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2 (1971) 189-208

7. Grosz, B., Sidner, C. : Attention, Intentions and the Structure of Discourse. Computational Linguistics 12 (1986) 175-204

8. Lindley, D. : Making Decisions, (2nd edition), Wiley (1985)

9. Quilici, A. Arguing about Planning Alternatives. Proc. 14th International Conference on Computational Linguistics. (1992) 906 – 910

10. Ramshaw, L. : A Metaplan Model for Problem-solving Discourse. Proc. 4th EACL (1989) 35 - 42, Manchester, UK.

11. Rao, A., Georgeff, M. Modelling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall (eds.) Proc. of the Second Conference on Knowledge Representation and Reasoning (1991) 473-484.

12. Searle, J.R. Speech Acts. Cambridge University Press (1969)

13. Sutton, R. , Barto, A. : Reinforcement learning: An introduction. The MIT Press. (1998)