MULTIPLE-AGENT PLANNING SYSTEMS

Kurt Konolige
Nils J. Nilsson

SRI International, Menlo Park, California

ABSTRACT

We analyze problems confronted by computer agents that synthesize plans that take into account (and employ) the plans of other, similar, cooperative agents. From the point of view of each of these agents, the others are dynamic entities that possess information about the world, have goals, make plans to achieve these goals, and execute these plans. Thus, each agent must represent not only the usual information about objects in the world and the preconditions and effects of its own actions, but it must also represent and reason about what other agents believe and what they may do. We describe a planning system that addresses these issues and show how it solves a sample problem.

INTRODUCTION

Certain tasks can be more advantageously performed by a system composed of several "loosely coupled," cooperating artificial intelligence (AI) agents than by a single, tightly integrated system. These multiple agents might be distributed in space to match the distributed nature of the task. Such systems are often called distributed artificial intelligence (DAI) systems [1]. We are interested here in systems where the component agents themselves are rather complex AI systems that can generate and execute plans, make inferences, and communicate with each other.

Among the potential advantages of such DAI systems are graceful (fail-soft) degradation characteristics (no single agent need be indispensable), upward extensibility (new agents can be added without requiring major system redesign), and communication efficiency (a message-sending agent can plan its "communication acts" carefully, taking into account the planning and inference abilities of the receiving agents).

In planning its actions, each agent must consider the potential actions of the other agents. Previous AI research on systems for generating and executing plans of actions assumed a single planning agent operating in a world that was static except for the effects of the actions of the planning agent itself. Examples of such systems include STRIPS [2], NOAH [3], and NONLIN [4]. Several important extensions must be made to planning systems such as these if they are to function appropriately in an environment populated by other planning/execution systems.

First, each agent must be able to represent certain features of the other agents as well as the usual information about static objects in the world. Each agent must have a representation for what the other agents "believe" about themselves, the world, and other agents. Each agent must have a representation for the planning, plan-execution, and reasoning abilities of the other agents. These requirements presuppose techniques for representing the "propositional attitudes" believe and want. Second, among the actions of each agent are "communication actions" that are used to inform other agents about beliefs and goals and to request information. Finally, each agent must be able to generate plans in a world where actions not planned by that agent spontaneously occur. We introduce here the notion of spontaneous operators to model such actions.

In this paper, we give a brief summary of our approach toward building DAI systems of this sort. It should be apparent that the work we are describing also has applications beyond DAI. For example, our multiple agents plan, execute, and understand communication acts in a manner that could illuminate fundamental processes in natural-language generation and understanding. (In fact, some excellent work has already been done on the subject of planning "speech acts" [5-6].) Work on models of active agents should also contribute to more sophisticated and helpful "user models" for interactive computer systems. The development of multiagent systems might also stimulate the development of more detailed and useful theories in social psychology--just as previous AI work has contributed to cognitive psychology. At this early stage of our research, we are not yet investigating the effects of differing "social organizations" of the multiple agents. Our work to date has been focussed on representational problems for such systems independent of how a society of agents is organized.

A MULTIPLE-AGENT FORMALISM

Each agent must be able to represent other agents' beliefs, plans, goals, and introspections about other agents. Several representational formalisms might be used. McCarthy's formalism for first-order theories of individual concepts and

propositions [7] is one possibility, although certain problems involving quantified expressions in that formalism have not yet been fully worked out. Another candidate is Moore's first-order axiomatization of the possible world semantics for the modal logic of knowledge and action [8-9]. Appelt [10] has implemented a system called KAMP that uses Moore's approach for generating and reasoning about plans involving two agents. We find Moore's technique somewhat unintuitive, and it seems needlessly complex when used in reasoning about ordinary (nonattitudinal) propositions. Here we develop a representation for each agent based on Weyhrauch's notion of multiple first-order theories and metatheories [11].

Using Weyhrauch's terminology, each computer individual is defined by the combination of a first-order language, a simulation structure or partial model for that language, a set of Facts (expressed in the language), and a Goal Structure that represents a goal for the agent and a plan for achieving it. We assume that each agent has a deductive system (a combination of a theorem-prover and attached procedures defined by the simulation structure) used for deriving new facts from the initial set and for attempting to determine whether goals and subgoals follow from the set of facts. Each agent is also assumed to have a planning system (such as STRIPS) for creating plans to achieve goals.

Using a typical "blocks-world" example, we diagram an agent's structure in the following way:

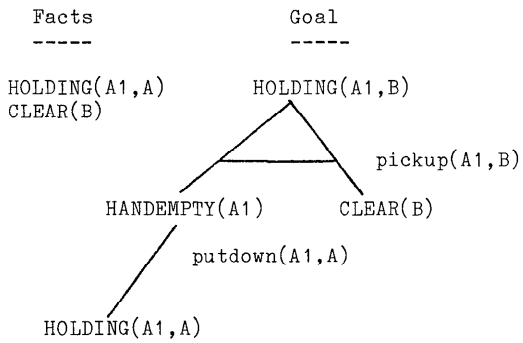
A1 (agent's name):

Facts	Goal
HOLDING(A1,A)	HOLDING(A1,B)
CLEAR(B)	

Viewed as a computational entity, an agent's structure is typically not static. Its deductive and sensory processes may expand its set of facts, or its planning system may create a plan to achieve a goal. Also, once a plan exists for achieving a goal, the agent interacts with its environment by executing its plan.

In this summary, we deal only with the planning processes of agents. In the example above, the occurrence of the goal HOLDING(A1,B) in A1's goal structure triggers the computation of a plan to achieve it. Once generated, this plan is represented in the goal structure of agent A1 as follows:

A1:



Plans are represented by goal/subgoal trees composed of planning operators and their preconditions. We assume a depth-first ordering of the operators in the plan.

Now let us introduce another agent, A0. Agent A0 can have the same sort of structure as A1, including its own first-order language, a description of the world by wffs in that language (facts), a simulation structure, a goal structure, and a planner and deducer. Some of A0's facts are descriptions of A1's structure and processes. By making inferences from these facts, A0 can reason about the planning and deductive activities of A1 and thus take A1 into account in forming its own plans. Also, a structure similar to A1's actual structure, and procedures similar to A1's deducer and planner, can be used as components of A0's simulation structure. Procedural attachment to these "models" of A1 can often be employed as an alternative method of reasoning about A1. (Of course, A0 may have an incomplete or inaccurate model of A1.)

Because A1's structure is a first-order language (augmented by certain other structures), A0 can use a formal metalanguage to describe it along the lines suggested, for example, by Kleene [12] and developed in FOL by Weyhrauch [11]. A0 has terms for any sentence that can occur in A1's language or for any of A1's goals or plans; the predicates FACT and GOAL are used to assert that some sentences are in A1's Facts list or goal structure. Consider the following example: assume A0 is holding block A, block B is clear, and A0 believes these facts and further believes that A1 believes A0 is holding A and that A1 believes B is not clear. A0 would have the following structure:

A0: Facts

HOLDING(A0,A)
CLEAR(B)
FACT(A1,'HOLDING(A0,A)')
FACT(A1,'~CLEAR(B)')

We use quote marks to delimit strings, which may have embedded string variables. The denotation of a ground string is the string itself. Thus, the intended interpretation of FACT(A1,'HOLDING(A0,A)') is that the wff HOLDING(A0,A) is part of the facts

list of A1 (that is, A1 "believes" that AO is holding A). By using the FACT predicate and terms denoting other agents and wffs, any facts list for other agents can be described. (AO can describe its own beliefs in the same manner.)

We purposely use "believe" instead of "know" because we are particularly interested in situations where agents may be mistaken in their representation of the world and other agents. In the above example, AO's opinions about its own and A1's belief about whether or not block B is clear are inconsistent. We avoid formalizing "know" and thus do not take a position about the relationship between knowledge and belief (such as "knowledge is justified true belief"). We can describe some of the usual properties of belief by axioms like FACT(x,p) => FACT(x,'FACT(x,p)'); i.e., if an agent believes p, it believes that it believes p. We do not, however, use an axiom to the effect that agents believe the logical consequences of their beliefs, because we want to admit the possibility that different agents use different procedures for making inferences. In particular, we want to emphasize that the deductive capabilities of all agents are limited.

While the static structure of A1 is described, for AO, by FACT and GOAL predicates, the action of A1's deductive system and planner can also be axiomatized (for AO) at the metalevel (see Kowalski [13] for an example). This axiomatization allows AO to simulate A1's deducer or planner by purely syntactic theorem-proving. Thus AO might use predicates such as ISPROOF(x,p) and ISPLAN(x,p) to make assertions about whether certain proof or plan structures are proofs or plans for other agents (or for itself).

In certain cases, AO can find out if A1 can deduce a particular theorem (or if A1 can create a plan) by running its procedural model of A1's deducer (or planner) directly, rather than by reasoning with its own facts. This is accomplished by semantic attachments of models of A1's deducer and planner to the predicates ISPROOF and ISPLAN in AO's metalanguage. Semantic attachment thus allows AO to "think like A1" by directly executing its model of A1's planner and deducer. (Here, we follow an approach pioneered by Weyhrauch [11] in his FOL system of using semantic attachments to data structures and programs in partial models.) The same kind of attachment strategy can be used to enable AO to reason about its own planning abilities.

The usual problems associated with formalizing propositional attitudes [8,14] can be handled nicely using the FACT predicate. For example, the atomic formula FACT(A1,'CLEAR(A) v CLEAR(B)') asserts the proposition that A1 believes that A is clear or B is clear, and is not confused with the formula [FACT(A1,'CLEAR(B)') v FACT(A1,'CLEAR(A)')], which asserts the different proposition that A1 believes that A is clear or A1 believes that B is clear. Furthermore, semantic attachment methods confer the advantages of the so-called "data base approach" [8] when appropriate.

Of particular importance among statements concerning AO's beliefs about A1 are those that involve "quantifying in," i.e., where a quantified variable appears inside the term of a FACT predicate. We follow the general approach of Kaplan [15] toward this topic. For example, the sentence (Ex)FACT(A1,'HOLDING(A1,x)') occurring among AO's facts asserts that A1 is holding an identified (for A1) block without identifying it (for AO).

AN EXAMPLE

We can illustrate some of the ideas we are exploring by a short example. Suppose that there are two agents, AO and A1, each equipped with a hand for holding blocks. Initially A1 is holding a block, A, and AO wants to be holding A. Suppose that AO believes these initial facts, but (to make our example more interesting) AO has no information about whether or not A1 itself believes it is holding A. Thus, the initial structure for AO is:

AO:	Facts	Goal
	HANDEMPTY(AO)	HOLDING(AO,A)
	HOLDING(A1,A)	

Let us assume the following planning operators (for both AO and A1). We use standard STRIPS notation [16]. ('P&D' denotes the precondition and delete lists; 'A' denotes the add list.)

```
putdown(x,b)    agent x puts
                block b on the table
  P&D:  HOLDING(x,b)
  A:  ONTABLE(b) & CLEAR(b) & HANDEMPTY(x)

pickup(x,b)     agent x picks up block b
  P&D:  CLEAR(b) & HANDEMPTY(x)
  A:  HOLDING(x,b)

asktoachieve(x,y,g)  agent x gives agent y the
  P:  T              goal denoted by string g
  A:  GOAL(y,g)

tell(x,y,s)   agent x tells agent y the
  P:  FACT(x,s)     expression denoted by string s
  A:  FACT(y,s)
```

Agents take into account the possible actions of other agents by assuming that other agents generate and execute plans to achieve their goals. The action of another agent generating and executing a plan is modelled by a "spontaneous operator." A spontaneous operator is like an ordinary planning operator except that whenever its preconditions are satisfied, the action corresponding to it is presumed automatically executed. Thus, by planning to achieve the preconditions of a spontaneous operator, a planning agent can incorporate such an operator into its plan.

Let us assume that agent AO can use the operator "achieve" as a spontaneous operator that models the action of another agent generating and executing a plan:

achieve(x,g) agent x achieves goal g by creating and executing a plan to achieve g.

PC: GOAL(x,g) & ISPLAN(x,p,g,x) & ISPLAN(x,p,g,AO)

D: **the delete list is computed from the plan, p**

A: FACT(x,g)
FACT(AO,g)

The expression ISPLAN(x,p,g,f) is intended to mean that there is a plan p, to achieve goal g, using agent x's planner, with facts belonging to agent f. Our precondition for achieve ensures that before AO can assume that condition g will be spontaneously achieved by agent x, AO has to prove both that agent x can generate a plan from its own facts and that agent x could generate a plan from AO's facts (to ensure that the plan is valid as far as AO is concerned).

Here are some axioms about ISPLAN that AO will need:

1) FACT(x,w) => ISPLAN(x,NIL,w,x)
(If x already believes w, then x has a plan, namely NIL, for achieving w.)

2) [ISPLAN(x,u,y,x) & PC(z,y) & OP(x,z,g)] => ISPLAN(x,extend(u,z),g,x)
(If x has a plan, namely u, to achieve the preconditions, y, of its operator, z, with add list containing g, then x has a plan, namely extend(u,z) for achieving g. The functional expression, extend(u,z), denotes that plan formed by concatenating plan z after plan u.)

The planning tree in Figure 1 shows a possible plan that AO might generate using its facts (including axioms about ISPLAN) and operators.

The sequence of operators in this plan is {tell(AO,A1,'HOLDING(A1,A)'), asktoachieve(AO,A1,'CLEAR(A)'), achieve(A1,'CLEAR(A)'), pickup(AO,A)}. Note that semantic attachment processes were used in several places in generating this plan. We leave to the control strategy of the system the decision about whether to attempt to prove a wff by semantic attachment or by ordinary syntactic methods. We are now in the process of designing a system for generating and executing plans of this sort. Space prohibits describing some additional features of our system, including its control strategy for generating plans. We plan to experiment with a complex of several agents, each incorporating planning systems like that briefly described here.

We gratefully acknowledge helpful discussions with Doug Appelt, Bob Moore, Earl Sacerdoti, Carolyn Talcott and Richard Weyhrauch. This research is supported by the Office of Naval Research under Contract No. N00014-80-C-0296.

REFERENCES

1. Sacerdoti, E. D., "What Language Understanding Research Suggests about Distributed Artificial Intelligence," in Distributed Sensor Nets, pp. 8-11. Paper presented at the DARPA Workshop, Carnegie-Mellon University, Pittsburgh, Pennsylvania (December 7-8, 1978).

2. Fikes, R. E. and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, 2(3/4), pp. 189-208 (1971).

3. Sacerdoti, E. D., A Structure for Plans and Behavior, (New York: Elsevier, 1977).

4. Tate, A., "Generating Project Networks," in IJCAI-5, pp. 888-893 (1977).

5. Searle, J. R., "A Taxonomy of Illocutionary Acts," in Language Mind and Knowledge, K. Gunderson (Ed.), (University of Minnesota Press, 1976).

6. Cohen, P. R. and C. R. Perrault, "Elements of a Plan-Based Theory of Speech Acts," Cognitive Science, 3(3), pp. 177-212 (1979).

7. McCarthy, J., "First Order Theories of Individual Concepts and Propositions," in Machine Intelligence 9, pp. 120-147, J. E. Hayes and D. Michie (Eds.), (New York: Halsted Press, 1979).

8. Moore, R. C., "Reasoning About Knowledge and Action," in IJCAI-5, pp. 223-227 (1977).

9. Moore, R. C., "Reasoning About Knowledge and Action," Artificial Intelligence Center Techincal Note 191, SRI International, Menlo Park, California (1980).

10. Appelt, D., "A Planner for Reasoning About Knowledge and Belief," Proc. of the First Annual Conference of the American Association for Artificial Intelligence, Stanford, California (August 1980).

13. Kowalski, R., Logic for Problem Solving, (New York: North-Holland, 1979).

14. Quine, W. V. O., "Quantifiers and Propositional Attitudes," in Reference and Modality, L. Linsky (Ed.), pp. 101-111, (London: Oxford University Press, 1971).

15. Kaplan, D., "Quantifying In," in Reference and Modality, L. Linsky (Ed.), pp. 112-144, (London: Oxford University Press, 1971).

16. Nilsson, N. J., Principles of Artificial Intelligence, (Menlo Park: Tioga Publishing Co., 1980).

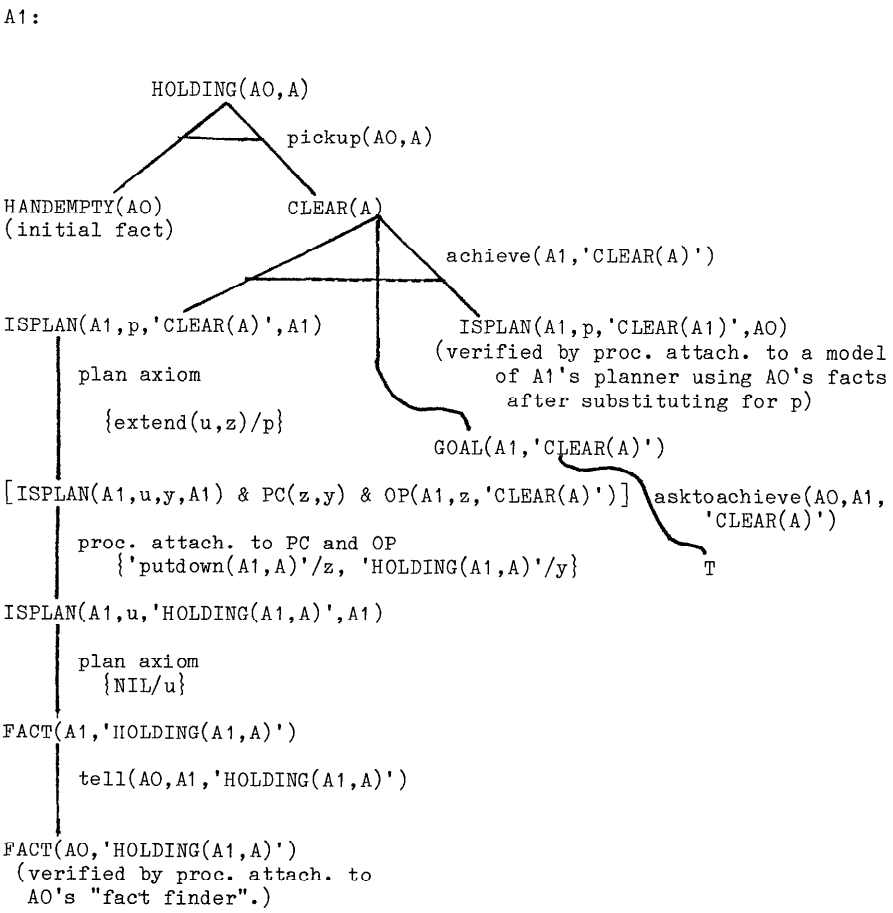


Figure 1