On Cooperation in a Multi-Entity Model (Preliminary Report)

Moshe Tennenholtz and Yoram Moses

Department of Applied Math, and Computer Science
The Weizmann Institute of Science
Rehovot 76100, Israel

Abstract

The problem of modeling a variety of domains within the framework of one general scheme is of central importance in AI. This paper presents the *Multi-Entity* model for multi-agent planning (MAP). This is a model in which a wide variety of scenarios can be represented, while many basic questions in MAP can be naturally represented and relatively efficiently computed. We use this model in order to study issues in MAP that we think are fundamental features of cooperation. These issues include: Achievement of cooperative goals, achievement of cooperative goals in the presence of failures, stable deals among agents (agents might not accept solutions that are irrational to them), tolerating agents that may deviate from a plan in order to improve their situation, and tolerating agents that might deviate in unknown ways from a plan (because they are liars, or because their plan has changed).

1. Introduction

The problem of modeling a variety of domains within the framework of one general scheme is of central importance in AI. An ideal mod$_e$'.ling scheme will be sufficiently expressive to describe many types of situations, will represent the world in a concise fashion, and will allow computing and analysing important features of the domain in an efficient manner.

Our focus in this paper is on domains that arise in multi-agent planning (MAP). In MAP, we are often dealing with robots or agents that act in a rather complex environment, may interact with each other, need to cooperate etc. Many experiments in multi-agent planning have been carried out in domains such as the "blocks world". In this domain the issue of representation and modeling poses rather little difficulty, and is greatly simplified due to the simplicity of the domain. However, it is never claimed that the blocks world can serve as a general model for multi-agent planning; rather, it is a simple domain in which to study various relevant aspects of the subject. A general framework often suggested for representing more complex environments for multi-agent planning is first-order logic. However, there are many respects in which first-order logic proves lacking for this purpose. For one, it does not provide a natural and efficient way of describing active and passive agents and interaction between agents. More importantly perhaps, there are no decidable, let alone efficient, general methods of computing central questions in MAP, such as whether goals of agents in a specific scenario are attainable (the tradeoff between expressiveness of modeling schemes and complexity is thoroughly discussed in [Lev86]). Thus, while first-order logic is quite expressive, its extreme generality hinders its usefulness.

This paper presents the *Multi-Entity* model for multi-agent planning, and uses it to suggest and analyse a number of general issues relevant to MAP. This model, which is based on modeling agents as sets of finite-state machines (fsms) is an intermediate model in which a wide variety of scenarios can be represented, while many relevant aspects of multi-agent scenarios can be naturally represented and relatively efficiently computed. In particular, we will show that STRIPS-style representations can be concisely modeled in our framework. The model presented in this paper does not model uncertainty; this is beyond the scope of this paper. In a related paper ([MT]) we show how our model can be modified in order to capture uncertainty. Models that describe multiple agent activity have been used in Theoretical Computer Science (e.g., [Lad79]), and in AI (e.g., [Geo86]), but none of them is especially suitable for issues in MAP.

Once the model is established, we introduce the *Cooperative Goal Achievement* problem (CGA), in which a set of agents try to collectively achieve all of their individual goals. We investigate the computational complexity of the appropriate decision problem. This

is shown to be decidable (but PSPACE-complete). We also treat the corresponding search problem (constructing a successful plan). Next, we relate the issue of faultiness from distributed systems to agents in the context of MAP. This is easily represented in our model, and again we study the CGA problem in this context. We treat the achievement of goals when failures may occur as a combinatorial game, and show that (perhaps surprisingly) the complexity of the appropriate cooperative goal achievement decision problem is the same as when no failures are possible. Next we define and investigate "stable solutions" for the achievement of goals in multi-agent systems. These are cooperative solutions that are suitable for rational deals, guaranteeing that no agent will deviate from the solutions. We define a variety of stable solutions, some of which are appropriate for the case of cooperative and honest agents that want to cooperate but will not accept an irrational deal, and some of which are appropriate for cases when agents may deviate from deals in order to improve their situation. This section exposes some connections between multi-agent planning and general concepts of Game Theory (for some other interesting connections between Game Theoretic concepts and MAP see [RG85], [GGR86]). Our "stable solutions" induce new cooperative goal achievement problems, which are also shown to be decidable. We then treat the case where non-truthful agents announce goals that are different from their actual ones. This treatment is appropriate in particular for robust multi-agent planning — when some agents' goals may change suddenly, etc.

The model presented in this paper is not necessarily the ultimate model for MAP. It may yet evolve further and improve. What is important is the spirit of this model, its expressiveness coupled with its relative tractability. The issues of MAP studied within the context of this model are fundamental features of cooperation in MAP. They deserve attention independently of the model; we present both the model and these concepts here in order both to show the applicability of the model, and to have a rigorous framework in which to study these aspects of cooperation.

Many details are omitted from this report due to lack of space. These details include the full formalism of the multi-entity model, and the full formalism of the definitions of stable solutions. The proofs of the theorems that appear in this paper are also omitted. All of these details appear in [TM].

2. The Multi-Entity Model

In this section we introduce our basic model (the full formal model appears in [TM]).

We represent the world (or a relevant part of it) as a multi-agent system. Some of the agents in our system are active, and correspond to humans, robots, etc. Other agents are passive and correspond to passive entities or objects in the world. Each agent is represented by a set of nondeterministic finite-state machines (fsms). Some of the fsms that constitute an active agent (and are called *active* fsms) can generate *basic actions* (or *actions).* All the fsms (of passive and active agents) are affected by these actions. The reader should note that we do not require that each active agent act as a fsm, but that the relevant physical description of an agent can be described by a finite number of states, and in each physical state the agent can perform one of a finite number of actions. The active agent itself may have very powerful computational abilities, which it may use for choosing which actions to perform in any given state. The internal state of the agent's computation need not be part of the representation. The reason that we use a set of fsms for describing an agent is the succinctness of representation. Agents are often composed of subparts that are fairly independent. By representing each such part (e.g., arms, legs, fingers) as a separate component we obtain a natural way of describing local actions and local change in an efficient manner.

A tuple of basic actions that are performed at a certain moment by the active fsms is called a *joint action.* Each joint action potentially affects the state of all the machines (both the passive and the active). The possibility of generating a basic action by a certain active fsm and the effects of a joint action on the state of a certain fsm may depend on the state of all of the fsms when the actions are performed (the general state of the world). There are no real limitations on what an "agent" should be. An agent can actually be a physical entity in the world (such as a hand, a door, etc.); it can also be composed of fsms whose states are a set of places where certain items may be, a set of values of an assertion, etc. A tuple of states of the fsms (a unique state for each fsm) is called a *configuration of the system.* Each fsm has an *initial state.* The tuple of initial states is called the *initial configuration of the system.*

A *run* of the system is a series of configurations $C_0, c_1, \ldots$ where C_0 is the initial configuration and for each t there is a joint action J_t that can be generated in c_t and changes the configuration of the system into c_{i+1}. Any sequence $c_0, J_0, c_1, J_1 \cdots$ that satisfies the above is called an *execution.* Finally, we assume that a joint action J that is performed in a configuration c determines a unique next configuration c'.

Discussion:

Modeling entities in a complex world by sets of fsms might seem a bit coarse at first. This, of course, depends on the required granularity of the representation. In many contexts we find that the role of inherently complex entities matters only in terms of a small number of abstract states. For example, the only relevant states of the weather on a summer day in London may be *rainy* or just *foggy* (and perhaps we should add *sunny* for the optimists). The complex physical structure underlying the causes of rain, fog, etc., will generally be irrelevant to the representation task.

The actual use of our model is very natural for many domains. We first have to identify the basic relevant entities (people, arms, doors, tools, etc.) that are involved in the problem. These entities will correspond to our agents. Each agent may now be decomposed into several fsms (for example, the fingers of a hand or the parts of the door). For every such machine we must specify its relevant states (places that a tool can be in, the position of an arm, etc.). Next, the actions each agent can generate in every given situation must be specified, as should the transitions induced by joint actions. Notice that the modularity of the model (many agents that are composed of many fsms) yields a succinct representation, and therefore the use of our multi-entity model is in many cases no less efficient than corresponding special-purpose representations. This modularity also enables us to perform modifications (add, delete or update) to the representation in a straightforward manner.

As an example of the expressiveness of our model consider a STRIPS-like (see [FN71]) representation in which the preconditions and postconditions (add-list/delete-list) are subsets of a set of atomic propositions and their negations. With each such atomic proposition we associate a passive agent that contains only one finite-state machine with two states — True or False (this describes whether the appropriate proposition is true or false). The robot's possible actions are represented by an active agent that consists of a single finite-state machine with one state. This finite-state machine can generate, for any possible action in the STRIPS-like representation, a unique basic action. The preconditions for the possible actions in the STRIPS-like representation become preconditions (on the states of the passive fsms) for the generation of the basic actions in our model. The postconditions (add list, delete list) are represented by adding appropriate transitions (joint actions) that affect the state of the passive fsms according to the delete list/add list in the STRIPS-like representation. Notice that the above simulation is efficient (the size of the STRIPS-like representation in our model is no larger than the original representation).

3. Cooperative Goal Achievement

In the previous section we introduced a general model where actions that are performed by a system of active agents and their effects on the world can be described. In this section we study aspects of cooperation in the context of this model The active agents (which we sometimes also refer to as robots) want to achieve certain goals. An active agent may sometimes be able to achieve its goal by itself, but at other times it cannot do without the help of other agents. Furthermore, the actions that are generated by one active agent may disturb another agent attempting to achieve its own goal. In fact, there can even be conflicts between different agents' goals.

In this paper we assume that the description of the world and the goals of the robots are common knowledge, and that any change in the world becomes immediately known to them. Our model is basically appropriate also for describing situations of lack of knowledge, and such treatment will appear in a pending paper ([MT]).

A *goal of an active agent* is a proposition in a propositional logic where the atomic propositions are the states of the fsms, and the connectives are -, A, V. The *Cooperative goal of the system* (denoted by G), is a tuple $G = (g_1 \ldots, g_m)$ where g_i $(1 < i < m)$ is the goal of the i-th active agent, and the number of active agents is m.

The *Cooperative goal achievement decision problem* (denoted by *CGA)* is defined as follows: given a world represented in our model, and a cooperative goal (7, is there a run $C_o, C_i, \ldots$ in the model that achieves all goals of G, or in other words satisfies

$$\forall j \exists i.\ c_i \models g_j$$

We remark that all of our results hold also when the goals are to be achieved simultaneously: $\exists i \forall j\ c_i \models g_j$

We can show:

Theorem 1: The "Cooperative Goal Achievement" decision problem is PSPACE-complete.

Corollary 2: If a cooperative goal can be achieved, then there is a finite execution that achieves it that can be computed and encoded in exponential space.

Let us consider a simple example of a situation involving cooperation and its representation in our model. There are three agents, Alice, Bob, and Chris,

standing on one side of a lake. There is a boat that requires at least two people to operate and can carry up to four. Alice and Bob want to go to place *X* on the other side of the lake, while Chris wants to go to Y, which is distant from *X*. In our model, we will represent each of Alice Bob and Chris as active agents, and the boat as a passive agent. One of the components of an active agent will be its location, and the possible actions will be getting into the boat, rowing in a particular direction, and getting off the boat. The boat can also be represented in terms of its location. Of course, in a more complex situation these representations can be refined much further. The joint actions are defined in such a way that if two agents row in the same direction, the boat moves in that direction. The model can now be completed in a straightforward manner. The agents' goals are now to be in their respective desired locations. It is easy to see that the CGA problem is solvable in this case: All agents row to Y, Chris gets off, and Alice and Bob continue to *X*. Notice that this solution requires the cooperation of Alice and Bob in helping Chris. They could attain their own goals by themselves, leaving Chris behind. We will return to this issue in Section 5.

4. Introducing Faultiness

Up to this point, the connections between Distributed Systems and Multi-Agent Planning were very general — the situation is by definition similar. Now, we would like to point to a specific concept in Distributed Systems that should often be considered in multi-agent planning — *faultiness.* Much of the research in the field of Distributed Systems is dedicated to fault-tolerance (see [DM] for example). Fault-tolerance is crucial in the context of MAP: in real life robots (active agents) might fail by crashing or by otherwise deviating from their normal behavior. We will now show the effects of crash failures on the achievement of cooperative goals in our model.

We capture "faultiness" in our model by allowing an active agent to fail at any step of an execution. We assume for now that the failures are *crash* failures (a robot that fails ceases to operate and is *never* revived), and that the number of robots that may fail in any execution is bounded above by t, where *t* is given and may depend on the number of robots — m. For each active agent we associate a distinguished state that we call *failure.* The failure of active agent number i given that the system is in configuration c occurs as follows: Every active fsm of active agent number i generates one of its possible actions in configuration *c* (the system will be changed according to this action), and then active agent number i moves into its *failure* state. When the agent is in its *failure* state all the fsms of this agent become passive, and no transition can change this distinguished "state". We assume that the failure of an agent is observable (as are all other changes in the world) by the other active agents.

The appropriate *CGA* will be changed into the question: is there a "strategy" for the robots against any failure pattern that ensures that all the non-faulty robots achieve their goals? We will not define "strategy" formally in this paper, and the reader can treat it as a strategy in a Combinatorial Game (where this is well-defined) between the active agents and the failures. Identifying whether a cooperative goal can be achieved in this framework may seem harder than before (we have to determine whether there is a *strategy* and not just "guess" an execution), but the reader should notice that there are cases where it becomes very easy (if it is known that achieving a certain goal requires the cooperation of more than $m - t$ active agents, for example). We can show:

Theorem S: The complexity of deciding whether a cooperative goal can be achieved by the non-faulty robots is PSPACE-complete both if (i) *t* is taken as input, and if **(ii)** $t = \frac{m}{2}$ (or in general $t = \alpha \cdot m$ for some fixed $0 < \alpha < 1$).

Corollary 4: If a cooperative goal where faultiness may appear can be achieved, then there is a finite "strategy" achieving it that can be computed and encoded in double exponential space.

Our example from Section 3 can easily be extended to the case where failures are possible. For example, an agent might break a hand and may therefore not be able to help operating the boat. Notice that the cooperative goal in the presence of one failure can not be achieved in this example. However, if there is another agent that wants to move to *X* then it becomes solvable in the presence of one failure.

5. Stability

In this section we introduce and define *stable solutions* to the Cooperative Goal Achievement problem. The solutions that appear in this section are executions (as in Section 3). We will be interested in the question of whether there is a solution in which all the active agents achieve their goals, and there is no "reason" for a subset of them to disagree with the solution, or to deviate from it. Such solutions are called *stable solutions.* We assume that each active agent wants to achieve his goal and to spend the least amount of resources. Other assumptions can be treated similarly. Therefore, a stable solution should be one where each active agent achieves his goal and

in which no set of active agents has reason to complain about irrational spending of resources or where no set of agents can change the solution into one in which they spend a smaller amount of resources. There are stable solutions that capture the idea of a rational deal. In these solutions active agents do not deviate from agreed solutions but they do not agree to solutions that are irrational for them. Other stable solutions capture the idea that active agents might deviate from the original plan in order to improve their situation, and therefore we look for solutions where no deviation is worthwhile, etc.

The amount of resources consumed by an active agent in our model is (roughly speaking) the number of actions the agent performs until it halts. More specific resources (e.g., time, money, oil) are not treated in this paper.

We now introduce the definitions of our stable solutions. More formal definitions appear in [TM|. First, we assume that the goals of the agents are common knowledge (they are part of the input of the problem). Afterwards, we will discuss the case where the goals are not common knowledge and the agents announce their goals (in this case we have to treat the case where agents can also lie about their actual goal).

- Trivial Stability: An execution e that achieves a cooperative goal is said to be a *stable solution according to the trivial stability criterion* if there is no solution *e'* that is at least as good as *e* for all agents and better than *c* for at least one agent. This solution captures the idea of rational solutions, where the active agents are fully cooperative.

- Rational Stability: An execution *t* that achieves a cooperative goal is said to be a *stable solution according to the rational stability criterion* if no subset of the active agents has a "strategy* that improves the situation of at least one of them and does not make the situation of any other agent in this subset worse, regardless of what the active agents outside this subset do. The meaning of "strategy* is as in Section 4, but now the game is between the above "subset of active agents" and the other active agents. Intuitively, this criterion says that there is no subset of the active agents for which performing *e* is a sacrifice.

- Stability against deviations: An execution *e* that achieves a cooperative goal is said to be a *stable solution according to the stability against deviations criterion* if no subset of the active agents can improve the situation of at least one of them without making the situation of the other agents in this subset worse, when the active agents *outside* this subset keep on performing the appropriate actions of e. This criterion captures the idea of a solution that no subset of the active agents have a reason to deviate from.

The following relations hold:

Theorem 5: *Stable solutions according to stability against deviations* $\subset$ *Stable solutions according to rational stability* $\subset$ *Stable solutions according to trivial stability*

The *CGA* solution presented in the example of Section 3 is stable according to the trivial stability criterion. No single agent can gain by deviating from this plan. However, recall that Alice and Bob can attain their goals and save resources if they leave Chris at the port of origin. Indeed, it is easy to see that no *CGA* solution for that example is stable according to rational stability. In [TM] we define and study additional notions of stability.

There is a similarity between some of the previous concepts and general concepts of stability that appear in Game Theory (see [Owe82], [LR57]). Our definitions can be thought of as having the spirit of Game-Theoretic concepts, but they are specially tailored for our framework. Our concepts are more qualitative than the corresponding concepts of Game Theory. Our agents want to achieve certain goals and minimise expenditure of resources. Therefore, we do not have to consider the set of all possible joint strategies as is done in Game Theory. Another important point is that our concepts are defined in the context of a model that describes change in a world. Transforming actions in our model to a general representation of a game is unnatural and results in stable solutions being much harder to compute (our treatment is exponentially better).

Given one of the above stability criteria, the question that is asked is whether we can characterize (give an algorithm for characterization) the systems and goals that have stable solutions according to this criterion, and construct such appropriate solutions. This is answered by the following general theorem:

Theorem 6: Any solution that obeys one of the above mentioned criteria is at most exponentially long (contains at most exponentially many joint actions until all the fsms generate no more actions).

Corollary 7: If a stable solution according to one

of the above criteria exists, then it can be computed and encoded in exponential space.

Up to this point, we assumed that the goals of the active agents are part of the input to the problems that we discussed. However, it is often the case that the identity of each agent's goal is obtained by the agent declaring what its goal is. In this case a malicious agent may lie by announcing a goal that is different from its actual goal, and then perhaps deviate from the joint plan in order to achieve its secret actual goal. A decent notion of stability in this case should require that the honest agents be guaranteed to attain their goals even despite such behavior by a restricted number of agents. More formally, we assume that there are *m* active agents, at most *t* of which are liars and might behave arbitrarily, while the others are honest and behave according to the given solution. We can now define:

> Stability against liars: A strategy for the active agents is called *stable against t liars* if it achieves all the goals in *G* of all the agents that do not deviate from it, given there are at most *t* such liars.

The reader should notice that the above ideas can also be used in other cases in which robust multi-agent plans are desired. For example, when we want our plans to be insensitive to a change of goals by a certain number of the active agents (in cases where announcing a change of goals is not possible or expensive).

Theorem 8: For any given *t* and any m-tuple *G* of announced goals, we can decide in exponential space whether there is a strategy that is stable against t liars. If such a strategy exists, then it can be computed and encoded in double-exponential space.

Acknowledgements

We would like to thank Uri Feige, Yishai Feldman, Rafi Heiman, Oded Maler, Muli Safra, Yoav Shoham, and A vital Shrift for helpful discussions during the different stages of this work.

References

[DM] C. Dwork and Y. Moses, Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures, to appear in Information and Computation, 1989.

[FN71] R. E. Fikes and N. J.Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence* 2, 1971, pp. 189-208.

[Geo86] M. P. Georgeff, Actions, Processes, and Causality, *Reasoning About Actions and Plans* (M. P. Georgeff and A. L. Lansky, eds.), Morgan Kaufmann, 1986.

[GGR86] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein, Cooperation Without Communication, *Proc. of AAA 1-86,* 1986, pp. 51-57.

[Lad79] R. E. Ladner, The Complexity of Problems in Systems of Communicating Sequential Processes, *Proc. 11th ACMSymp. on Theory of Computing*, 1979, pp. 214-223.

[Lev86] H. Levesque, Knowledge Representation and Reasoning, *Ann. Rev. Comput. Sci* 1, 1986, pp. 255-287.

[LR57] R. D. Luce and H. Raiffa, *Games and Decisions- Introduction and Critical Survey,* John Wiley and Sons, 1957.

[MT] Yoram Moses and M. Tennenholts, Planning in Uncertain Territory Using a Multi-Entity Model. unpublished manuscript, 1989.

[Owe82] G. Owen, *Game Theory (2nd Ed.),* Academic Press, 1982.

[RG85] J. S. Rosenschein and M. R. Genesereth, Deals Among Rational Agents, *Proc. 9th International Joint Conference on Artificial Intelligence,* 1985, pp. 91-99.

[TM] M. Tennenholts and Yoram Moses, On Cooperation in a Multi-Entity Model Technical Report, Weizmann Institute, 1989.