# Planning to Please: Following Another Agent's Intended Plan

**Eithan Ephrati**
**Jeffrey S. Rosenschein***
Computer Science Department
Hebrew University
Givat Ram, Jerusalem, Israel
tantush@cs.huji.ac.il, jeff@cs.huji.ac.il

**Abstract**

In this paper we analyze a particular model of control among intelligent agents, that of *non-absolute control*. Non-absolute control involves a "supervisor" agent that issues orders to a "subordinate" agent. An example might be a human agent on Earth directing the activities of a Mars-based semi-autonomous vehicle.

Both agents operate with essentially the same goals. The subordinate agent, however, is assumed to have access to some information that the supervisor does not have. The agent is thus expected to exercise its judgment in following orders (i.e., following the true intent of the supervisor, to the best of its ability). After presenting our model, we discuss the planning problem: how would a subordinate agent choose among alternative plans? Our solutions focus on evaluating the distance between candidate plans, and is appropriate to any scenario in which one agent wants to follow (as much as possible) another agent's plan.

**Key Words:** Planning, communication, multiple agents, non-absolute control, plan deviation

---

*This is a revised and expanded version of the paper "Constrained Intelligent Action: Planning Under the Influence of a Master Agent," that appeared in the Tenth National Conference on Artificial Intelligence, San Jose, California, July 1992, pages 263–268.

# 1 Introduction

Imagine that you have sent your robot Explorer to analyze the Martian surface. Though your communication with him is limited, you are able, from time to time, to send him general goals, such as "Get a soil sample from Region $A$." Explorer is an obedient robot, but after failing to hear from him for some time, you ask why there's been a delay in retrieving the soil. Explorer informs you that because there is a large valley between his current location and Region $A$ (a valley of which you, his master, had been unaware), he has begun construction of a small plane to fly over the valley. In truth, getting the soil sample wasn't that important to you. Had you known the cost of Explorer's plan to accomplish your goal, you would have been willing to have him get a sample from Region $B$. Explorer *knew* that you didn't know about the valley, so he could have reasoned that your plan for the soil collection was much simpler than his. If only he had had the intelligence to inform you of the discrepancy!

On another occasion, you tell Explorer to move to Region $C$. While you believe he is able to do this using a particular route, he knows of two alternative shortcuts. Taking either shortcut will cause side effects (one route may disturb a scientifically interesting crater in the region and the second route might cause damage to his photoelectric cells). By what criteria should Explorer plan his travels, given his knowledge of your goals?

## 1.1 Multi-Agent Planning

Research on planning in Distributed Artificial Intelligence (DAI) has focused on two major paradigms: *planning for multiple agents* and *distributed problem solving*. In the first paradigm, a single intelligent agent (the "master") constructs a plan to be carried out by a group of agents (the "slaves"), then hands out pieces of the plan to the relevant individuals [Rosenschein1982, Konolige1982, Lesser1987, Katz and Rosenschein1993]. In the second paradigm, a group of intelligent agents jointly construct the final plan, and subsequently cooperate in carrying it out [Genesereth *et al.*1986, Durfee and Lesser1987].

In the master-slave model, the slave follows the exact detailed orders of the master. It has no interests of its own, and does not attempt to reason about the master's knowledge or plans. In the second model, all agents may share equally in control and decision making. Each agent might have its own goals and control its own actions. To promote its interests, each can reason about other agents' knowledge and goals. In some scenarios, agents have conflicting goals, negotiate, and compete over resources [Kraus *et al.*1991, Zlotkin and Rosenschein1991].

We are interested in that important family of scenarios in which the control method combines both aspects. In these scenarios, there are *supervising* and *supervised* agents, where the control of a supervising agent over the supervised one is non-absolute. Such a hierarchy is typically exhibited in organizations with a pyramid structure, but also characterizes all other situations in which the supervised agent is expected to show some intelligence. We will refer to the supervising agent as the "supervisor," and to the supervised agent simply as the "agent."

The approach explored in this paper is novel in that it makes use of the varying perspectives of the supervisor and of the agent to constrain the actions of the latter. The differing knowledge held by the supervisor and the agent need not, in all cases, be reconciled; sometimes it is enough for the automated agent to consider their different viewpoints and adjust his actions accordingly. There is an element of "cognitive influence" operating between agents: the supervised agent uses his model of the supervisor to constrain his own activities. The techniques suggested in this paper may serve in any scenario where one agent is interested in following (as much as possible) another agent's plan.

## 1.2 Related Work

The subject of non-absolute control (i.e., how an agent can coherently integrate local goals with outside goals) has received some attention in DAI and related fields. Here, we briefly mention some of this work.

Durfee [Durfee1988] has looked at the problem in the domain of distributed vehicle sensing. His method of "partial global plans" (PGP) allows agents to coordinate activity around common tasks. The agents may exist at the same level of an authority hierarchy, or one may be above another in that hierarchy. Agents do not, however, reconcile their activity by measuring distance between candidate partial global plans. In addition, all local plans that fully satisfy the constraints imposed by a PGP are satisfactory; in our approach, we derive useful information from plan deviation, even though all plans might be technically correct (i.e., achieve the goals).

Sycara has been interested in a somewhat similar problem within the framework of negotiation. Her method of "persuasive argumentation" is based upon modeling another agent's plan so as to find a way to influence ("convince") that agent to cooperate [Sycara1988, Sycara1989]. Sycara considers interactions with potential conflict, such as those between labor and management in a wage negotiation. In our model, however, there is no conflict between the goals of the supervisor and supervised agent. The agent is interested in following the desires of the supervisor, but needs to instantiate those desires given his (the agent's) own, more accurate, information about the environment. We are mainly concerned with how to have the agent act efficiently, without full transmission of data and plan steps back and forth between supervisor and agent.

Malone has also been interested in issues relating to the organization of agents (both human and artificial), and has published various analyses of different human organizations and their relationship to machine organization [Malone1986]. In [Malone *et al.*1988], Malone discusses the Enterprise system, which uses an expanded version of the Contract Net to allocate tasks in a distributed system. The agents in that architecture have the ability to reason about whom to send tasks to, and which tasks to accept, but cannot reason about another agent's knowledge and adjust their actions accordingly. In our model, it is precisely this latter kind of reasoning with which we are concerned.

In economics, the subfield of principal-agent theory addresses a problem somewhat complementary to the one in which we are interested. There, the supervisor (principal) tries to find the right incentives (payment) that will make the agent report true data about the domain in which it operates and, thus, in effect, to inform the principal how to make him more productive [Holmstorm and Milgrom1991].

In the field of telerobotics, there has also been discussion of how to effectively manipulate robots at a distance, though robots there are generally less intelligent than those in which we are interested (see, for example, [Conway *et al.*1990]).

# 2 Determining Plan Deviation

We are concerned with the following possible control relationships between the supervisor and the agent:

1. The supervisor generates and sends the agent a complete plan;

2. The supervisor does not (or cannot) generate a complete plan, but generates and sends the agent a partial (abstract) plan;

3. The supervisor has not generated a complete (or partial) plan, or perhaps has generated it but cannot send it because of constraints on communication. Instead, the supervisor sends a high-level goal.

Given a complete plan, the agent should in principle follow it, but this may not be possible or desirable if the supervisor's model of the world is faulty. Thus, even in this situation, or when the agent has been given a partial plan or a goal, the agent is expected to generate a complete plan based on its own, more accurate knowledge. The agent's complete plan should not differ radically from the supervisor's specified plan, or (if no complete plan was specified), what the agent predicts the supervisor *would* have generated as a plan.

The intuition is that if the world does not differ greatly from what the supervisor believes, and if the agent's model of the supervisor's beliefs is accurate, the plans they generate will also not differ greatly. If there is sufficient deviation between their plans, then someone's model is wrong, and some action should probably be taken (e.g., communicate). This is true when the the agent's plan is much "better" than the supervisor's plan, and when it is much "worse"; in both cases, some realignment may be necessary. Thus, we are concerned with measuring the distance between plans.

Note that we consider plan deviation among correct plans; even though they all achieve the goal, some are more suitable than others. We are concerned with more than just having a plan to achieve a goal—we are concerned with the structure of the resulting correct plan.

The main research issues of concern in these situations are reasoning about knowledge, nonmonotonic reasoning, shared knowledge and knowledge revision, communication, and planning. In this paper, we limit ourselves to issues of planning. A more detailed discussion of the other issues may be found in [Ephrati and Rosenschein1991].

## 2.1 A Blocks World Example

Consider a scenario in the slotted blocks world as described in Figure 1. This example will be used several times in the paper. The domain is described

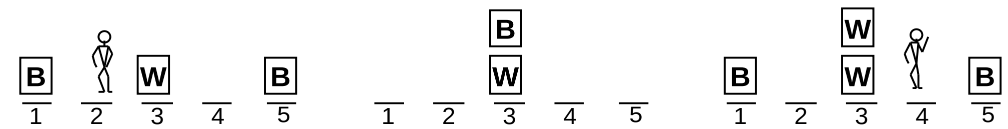


Figure 1: A scenario in the slotted blocks world

by the following predicates: **Blocks**$(n)$ — The total number of blocks in the world; **Stacked**$(block1,\ block2,\ slot)$ — $block1$ is on top of $block2$, and located at $slot$ (if $block1$ is on the table, $block2$ and $slot$ are identical); **At**$(s)$ — The agent is at slot $s$. The domain operators are: **Go**$(s_i, s_j)$ — The agent goes from $s_i$ to $s_j$; **Carry**$(b, s_i, s_j)$ — Agent carries block $b$ from $s_i$ to $s_j$; **Paint**$(b, color)$ — Agent paints block $b$ a particular color; **Destroy**$(b)$ — Agent destroys block $b$; **Create**$(b)$ — Agent creates a new block $b$.

In the initial state the agent believes the supervisor's knowledge of the world to be described by the following ($B$ refers to any Black block, $W$ to any White one): $\{Blocks(3)$, $Stacked(B, s_1, s_1)$, $At(s_2)$, $Stacked(W, s_3, s_3)$, $Stacked(B, s_5, s_5)\}$.
The agent's own knowledge of the world is:
$\{Blocks(4), Stacked(B, s_1, s_1), At(s_4), Stacked(W, s_3, s_3), Stacked(W, W, s_3), Stacked(B, s_5, s_5)\}$.
Given only the supervisor's goal, $\{Stacked(W, s_3, s_3),\ Stacked(B, W, s_3)\}$, the agent must decide which of the many alternative plans for achieving the goal he should carry out.

## 2.2 Cost Function and Primacy of Action

When we speak of any agent's being "autonomous," we are referring primarily to the degree to which it can take independent action in the domain. This fact implies that one key issue is how the supervisor can *predict* and *restrict* the supervised agent's actions. This observation in turn suggests the usefulness of a Cost Function over actions. There are several important uses for such a cost function:

- Providing a safety net over a supervised agent's actions; even if the actions are in pursuit of a goal, they can be restricted so as to be within a cost limit.

- Reflecting irreversibility of actions; some of the actions that the agent can take may be critical in the sense that they cannot be undone. Obviously such actions should be identified by both the agent and its supervisor. A cost function is one possible way to identify such actions [Finger1986].

- Providing a flexible control mechanism; since a cost function provides a quantitative measurement of the agent's activities and since these activities characterize the agent's autonomy, limiting the overall cost of these activities can serve as a flexible control handle (see Section 3).

To serve these purposes, a good function should reflect the real effort and resources invested in performing the action and the difficulty of reversing it. As such, it should preferably be dynamic and modifiable by both agents.

## 2.3 Assumptions and Definitions

- The agent keeps a model of the supervisor's knowledge base (KB) concerning the domain in which the agent operates, and updates the model during the interaction. This model may in fact differ from the actual view of the domain held by the supervisor.

- The *goal* $G$ is a set of predicates that might be given by the supervisor to the agent. $S_G$ are the states of the world in which G holds; we use $s_G$ to stand for any state in $S_G$.

- An agent $a$'s plan $P(a, s_0^a, s_G^a)$ (denoted by $P^a$) is a set of consecutive operations: $[op_1^a, op_2^a, \ldots, op_n^a]$, that brings about $s_G^a$ starting from the initial state $s_0^a$. The set of all such alternative plans is $\mathcal{P}^a$.

- Each operation of a plan $P$ transforms the world from one state into another $[s_0, s_1, \ldots, s_{n+1}]$ (such that $s_{n+1} = s_G$).

- Given a *cost function* ($C$: $OP \rightarrow \mathbb{R}$) over the domain's operators [Finger1986], we define the cost of a plan $C(P)$ to be $\sum_{k=1}^{n} C(op_k)$.

Based on a comparison with the supervisor's complete plan (either transmitted, or generated by the agent from a goal or partial plan using its model of the supervisor's knowledge base), the agent has to choose one plan from $\mathcal{P}^a$. In fact, the generation of the plans in $\mathcal{P}^a$ can themselves be guided by the supervisor's plan $P^s$, and the entire set $\mathcal{P}^a$ may not need to be explicitly generated.

Assume the following cost function over the operators of our example: $C(Go(s_i, s_j)) = \mid i - j \mid$, $C(Carry(b, s_i, s_j)) = 2\times \mid i - j \mid$, $C(Paint(b, color))$ = 6 if *color* is Black and 8 if it is White, $C(Destroy(b)) = 3$ and $C(Create(b))$ = 18.

In the example of Figure 1, the assumed supervisor's plan is $\langle Go(s_2, s_1)$, $Carry(B, s_1, s_3)\rangle$ $[cost 5]$. The agent is faced with choosing one of the following alternative plans (for simplicity, we ignore some other possibilities):

$$
\begin{aligned}
P_1^a &= \langle Go(s_4, s_3),\ Paint(W, Black)\rangle[cost 7]\\
P_2^a &= \langle Go(s_4, s_3),\ Destroy(W),\ Go(s_3, s_1),\ Carry(B, s_1, s_3)\rangle[cost 10]\\
P_3^a &= \langle Go(s_4, s_3),\ Destroy(W),\ Go(s_3, s_5),\ Carry(B, s_5, s_3)\rangle[cost 10]\\
P_4^a &= \langle Go(s_4, s_3),\ Carry(W, s_3, s_4),\ Go(s_4, s_5),\ Carry(B, s_5, s_3)\rangle[cost 8]\\
P_5^a &= \langle Go(s_4, s_3),\ Carry(W, s_3, s_2),\ Go(s_2, s_1),\ Carry(B, s_1, s_3)\rangle[cost 8].
\end{aligned}
$$

What follows are some plausible criteria for measuring distances between plans. Although we refer here to the entire plan, it may be possible to take into consideration *abstract* plans (i.e., to measure plan distances at higher levels of abstraction, incompletely specified), so as to reduce the computational complexity. For example, we might consider only operations that bring about a "necessarily true proposition" as defined by Chapman [Chapman1987]. Another motivation for working with abstract plans may be that this is, in fact, the suitable level at which to relate to the supervisor's plan.

# 3 Comparing Outcomes and Costs

One way of evaluating the distance between plans is to take into consideration only the final outcome of a plan or its overall cost. The following are three alternative ways of doing so:

**(1) Comparing the cost of plans generated by the agent to the cost of the supervisor's plan:** One method of comparison would be to consider only the cost of each plan, regardless of its actual steps or side effects. Such a consideration suggests the following cost metric:

$$D_c(P_1, P_2) = \mid C(P_1) - C(P_2) \mid .$$

Under certain circumstances, it might be satisfactory to choose the plan that minimizes the cost difference from the supervisor's (assumed) plan, or any plan such that $D_c(P, P^s)$ does not exceed a freedom threshold parameter $T_c$. Using the cost function given above, we get the following distance values: $D_c(P_1^a, P^s) = 2$, $D_c(P_2^a, P^s) = 5$, $D_c(P_3^a, P^s) = 5$, $D_c(P_4^a, P^s) = 3$, $D_c(P_5^a, P^s) = 3$. With the criterion of cost difference minimization, $P_1^a$ would be chosen.

**(2) Comparing utilities:** If possible, it might in fact be preferable to consider the supervisor's and agent's utility functions over plans (actions and effects), not simply their cost functions over actions. The notion of utility can capture much more information about the supervisor's preferences than the notion of simple cost. We would then compare the supervisor's utility resulting from the agent's alternative plans with the utility resulting from the supervisor's plan. Using utility relates to the approach taken in [Ephrati and Rosenschein1992]. We could define a utility metric as follows (where $u_s$ stands for the supervisor's utility resulting from a given plan):

$$D_u(P^s, P^a) = \mid u_s(P^s) - u_s(P^a) \mid .$$

If, for example, it is known to the agent that the supervisor relates some utility gain to the fact that slot 1 is clear, $P_2^a$ would become preferred over $P_3^a$, though their costs are identical.

Given the existence of a utility function for the supervisor, the agent might in fact take the simpler route of just maximizing the supervisor's utility. This assumes, of course, that a sufficiently accurate model of the supervisor's utility is available. That is, to realize what the supervisor's utility of a plan is, the agent would need to have all the supervisor's subgoals (implicitly achieved by its intended plan) along with their utility, and the utility of any side-effects caused in the environment. This is a much greater demand than the simple cost information we generally use in this paper.

**(3) Comparing the deviation of the KBs:** The freedom the agent has in generating plans may lead to some unpredicted side effects, and there is the potential that these side effects are undesirable from the supervisor's point of view. One symptom of such side effects is deviation between the supervisor and agent KBs. An intuitive solution is, therefore, to have the agent choose actions so as to minimize KB deviation. One method of evaluating the distance between KBs is to consider the cost of the plan that would transform one into the other. Using this criterion, the agent might be expected to minimize (or bound to $T_o$) the following "outcome metric":

$$D_o(P^a, P^s) = \max(C(P(a, s^a_G, s^s_G)), C(P(s, s^s_G, s^a_G)))$$

Applying the outcome metric to the example we get the following values: $D_o(P^a_1, P^s) = 22$, $D_o(P^a_2, P^s) = 0$, $D_o(P^a_3, P^s) = 12$, $D_o(P^a_4, P^s) = 32$, $D_o(P^a_5, P^s) = 20$. $P^a_2$ minimizes KB deviation. For example, the goal state that is created by $P^a_1$ differs from the one that is generated by $P^s$ by $\{Blocks(4), Stacked(B, s_1, s_1)\}$. The plan that transforms the world from the agent's actual goal state to the supervisor's presumed goal state is $\langle Go(s_3, s_1)$, $Destroy(B)$, $Go(s_1, s_3)\rangle[cost7]$. The plan that transforms the world from the supervisor's presumed goal state to the agent's actual goal state is $\langle Go(s_3, s_1)$, $Create(B)$, $Go(s_1, s_3)\rangle[cost22]$. Thus, the cost of the more expensive plan is 22.

A more sophisticated outcome metric would distinguish between differences caused by the plan, and differences that preceded plan execution. In the example above, there was already a difference in the number of blocks in the world prior to the plan's execution, and the outcome metric should perhaps not consider the variation in number of blocks at plan's end.

**(4) Avoiding irreversible consequences:** The solution presented above might not always be compatible with the kind of intelligence we would like the agent to exhibit. Consider a user (the supervisor) ordering "**rm** `~/pub/*`" to an intelligent UNIX operating system agent (the agent) [Finger1986]. If the agent is aware that the user does not know about the existence of some files in his `/pub` directory, we would like it to warn the user, although the deletion of all files in this case would actually bring the two KBs closer.

Situations can thus occur where the KBs get closer, but there is a loss of resources known only to the agent. Therefore, we want to prevent the agent from changing important parts of the world that are not known to the supervisor ($s_0^a - s_0^s$). This could be done by bounding the cost of the plan needed to recreate the resources:

$$C_r(P^a, P^s) = C(P(a, s_G^a, s_0^a - s_0^s)).$$

Notice that this constraint can be integrated into the cost metric if the cost of an operation reflects the cost of reversing it. This is not, however, always desirable since (for instance) we would not want the **rm** operation to be very expensive under all circumstances.

In our example, the difference between the two knowledge bases is: $\{Blocks(4), At(4), Stacked(W, W, s_3)\}$. Therefore $C_r(P_1^a, P^s) = 9$, $C_r(P_2^a, P^s) = 22$, $C_r(P_3^a, P^s) = 22$, $C_r(P_4^a, P^s) = 7$, $C_r(P_5^a, P^s) = 7$. For instance, to restore his exclusive knowledge from the final state of $P_1^a$, the agent would perform $\langle Paint(B, White), Go(s_3, s_4)\rangle$. Starting at the final state of $P_2^a$, the agent would perform $\langle Carry(B, s_3, s_2),\ Go(s_2, s_3),\ Create(W),\ Go(s_3, s_4)\rangle$. $P_4^a$ and $P_5^a$ minimize the cost needed to recreate resources.

# 4 Distance Between Sets of States

Above, we treated each plan as a single, indivisible unit. Alternatively, we could take into consideration in our evaluation *each step* of a plan, and the corresponding state to which it leads. These states can be used to measure distance between the plans. We associate with two plans $(P_1, P_2)$ their corresponding sets of states $(S^{P_1}, S^{P_2})$. Figure 2 shows the corresponding sets of states of $P_1^a$, $P_3^a$, $P_5^a$ and $P^s$. Assuming the existence of a distance metric between two states $d(s_1, s_2)$ (see the following section), the agent could make geometric measurements of distance between *sets* of points adequate for measuring the difference between two plans. Here are three plausible metrics between sets of points.

**(1) The Hausdorff metric:** One way of measuring inter-plan distance is to choose one "representative" state from each plan, and consider their distance. Above, for example, we considered the plans' *final* states to be representative. Other alternatives might be to take the plans' maximally distant states (or minimally distant) as representative, or the two "centers of gravity" of both plans.

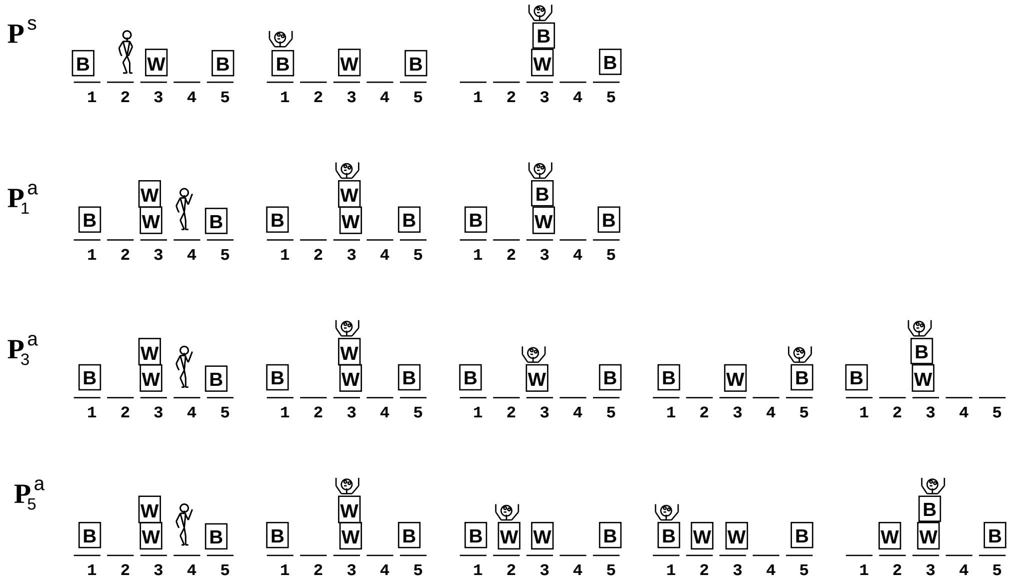

Figure 2: Sets of states corresponding to plans

The Hausdorff metric [Atallah1983, Arkin *et al.*1990] is also based on the distance between a representative state from each plan. This metric is used in the field of image processing to measure the distance between two shapes. Taking the state in the first set that is closest to any state in the second set, one measures the distance between this state in the first set and the *farthest* state in the second set. Doing this in both directions (both plans as the first set), one takes the maximal distance: $H_{max}(S^s, S^a) = \max(h(S^s, S^a), h(S^a, S^s))$, where $h(A, B) = \max_{a \in A} \min_{b \in B} d(a, b)$. It may be preferable at times to use the sum, $H_{sum}$, of the $h$ metrics.

Assume that the inter-state metric returns the following distance values between each of the states in the supervisor's plan, and each of the states in the agents' plans (as described in Figure 2). For simplicity, we look at only three plans for the agents. The listing below compares, for example, the distance between the first state of the supervisor's plan, $s_0$, and the 3 states in $P_1^a$, the 5 states in $P_3^a$, and the 5 states in $P_5^a$:

$(s_0^s$ —$)$ $(\langle 5, 4, 4\rangle, \langle 5, 4, 1, 3, 5\rangle, \langle 5, 4, 3, 4, 8\rangle)$,
$(s_1^s$ —$)$ $(\langle 6, 5, 5\rangle, \langle 6, 5, 2, 4, 6\rangle, \langle 6, 5, 4, 3, 9\rangle)$,
$(s_2^s$ —$)$ $(\langle 8, 7, 3\rangle, \langle 8, 7, 4, 6, 8\rangle, \langle 8, 7, 8, 9, 3\rangle)$.

These values yield $H_{sum}(S^{P^s}, S^{P_1^a}) = 10$, $H_{sum}(S^{P^s}, S^{P_3^a}) = 9$, $H_{sum}(S^{P^s}, S^{P_5^a}) = 8$. $P_5^a$ minimizes the Hausdorff metric.

**(2) The "Shifting" distance:** This measurement considers the overall differences between two plans. The measurement is done by summing the distance of each state in one plan from the set of states in the other. Defining the distance between one state and a set of states to be $D(s', S) = \min_{s \in S} d(s', s)$, we get the distance between one set of states and another to be the total shifts needed for merging this set with the other:

$$Sh(S^a, S^s) = \sum_{s \in S^a} D(s, S^s).$$

The distance between the sets can be defined by the *min*, the *max*, or, more informatively, the summation of $(Sh(S^a, S^s), Sh(S^s, S^a))$. As with the use of an "outcome metric," this measurement suffers from the fact that as the states converge the distance gets smaller, even though important data may be lost.

Following the values given in the example above, we get: $Sh_{sum}(S^{P^s}, S^{P_1^a}) = 24$, $Sh_{sum}(S^{P^s}, S^{P_3^a}) = 25$, $Sh_{sum}(S^{P^s}, S^{P_5^a}) = 27$. $P_1^a$ minimizes the Shifting Distance metric.

**(3) The "Dynamic deviation" distance:** This metric considers the sequential order in which the states of each plan were generated. Let $S^s = \{s_0^s, s_1^s, \ldots, s_m^s\}$ and $S^a = \{s_0^a, s_1^a, \ldots, s_n^a\}$. Assuming (without loss of generality) that $m = \min(m, n)$, we define the distance between the plans to be the summation of the sequential relative deviations produced by operators of each plan:

$$D_s(S^a, S^s) = \sum_{i=1}^{m} d(s_i^a, s_i^s) + \sum_{i=m}^{n} d(s_i^a, s_m^s).$$

Referring to the example, this metric yields: $D_s(S^{P^s}, S^{P_1^a}) = 13$, $D_s(S^{P^s}, S^{P_3^a}) = 28$, $D_s(S^{P^s}, S^{P_5^a}) = 30$.

The deviation metric captures, in a sense, the "direction" of the deviation (i.e., whether it is growing or shrinking). The deviation metric's advantage with respect to the other two metrics is its relatively easy computation. Using the Hausdorff metric is preferable when we are concerned only in preventing the agent from radical exceptions to the supervisor's intended plan, while not being bothered by many small variations. The Shifting metric, on the other hand, is suitable when the overall distance from the intended plan (whether caused by one drastic change or many small ones) is of importance.

A computational advantage of the deviation metric and of the Hausdorff metric is that they can better constrain the generation of a plan dynamically. For example, the Hausdorf metric considers only one member of $S^a$ while the Shifting measurement takes all of them into account.

All these metrics (or their variations) may be used with different freedom parameters. For example, using the Hausdorff metric, if our agent plans "forward" and follows the "best first" strategy, after generating the first two states of the three plans it would choose $P_5^a$ and will not find it necessary to further explore the two other alternatives, because the distance to them is already too large.

# 5 Metric Between States

Our techniques above depended on our being able to provide a distance metric between states. There are several sensible ways to define such a distance metric, and the following are some basic ideas. Any of the following state metrics can be used as part of any of the plan metrics mentioned above.

**(1) Distance between predicates:** Since we consider each state to be described by a set of predicates, a straightforward metric can be generated by the summation of the differences between conflicting predicates of the two states in question (this is related to the measures, such as cardinality of sets, proposed by Ginsberg for evaluating distance between possible worlds in his research on counterfactual planning [Ginsberg1986]). As a trivial example, consider two states that differ only by the location of one object ($obj$). The definition $\mathit{Diff}(At(obj, x_1, y_1),\ At(obj, x_2, y_2)) = \sqrt{\mid x_1 - x_2 \mid + \mid y_1 - y_2 \mid}$ is sufficient to serve as a distance metric between these states. This method can be refined by giving different weights to such metrics in accordance with some special considerations. For example, if there is a dangerous region in the agent's operating domain the above metric should be given a higher weight if $(x_2, y_2) \in DangerSet$. In that case the location might be used to change the weight of other differences such as $\mathit{Diff}(Armed(obj), UnArmed(obj))$.

The values that were used as the metric between states in the previous examples were derived from the following definitions (based on the cheapest operation that can eliminate the difference): *Diff*($Blocks(n_1)$, $Blocks(n_2)$) $= 3\times \mid n_1 - n_2 \mid$, *Diff*($At(s_i)$, $At(s_j)$) $= \mid i - j \mid$, *Diff*($Stacked(b, loc, s_i)$, $Stacked(b, loc, s_j)$) $= 2\times \mid i - j \mid$.

**(2) Considering ultimate goals:** A more meaningful metric might consider either the agent's or the supervisor's global goals (or both). For example, if one of the supervisor's global goals is *Above*($obj_1$, $obj_2$) and both states satisfy $Stacked(obj_3,\ obj_2)$, we would like *Diff*($Stacked(obj_1,\ obj_3)$, $Stacked(obj_1, obj_4)$) to return a higher value than, for instance, *Diff*($Stacked(obj_1,\ obj_5)$, $Stacked(obj_1,\ obj_4)$). Global goals can be given at a higher level of abstraction, such as "$\min_{obj \in U.S.Navy} In(obj, Iraq)$." The fact that such ultimate goals can be from different sources (i.e., from the supervisor and agent) allows the consideration of subjective points of view (with respect to these sources) and thus enriches the overall control mechanism. If the supervisor of the example had as a global goal the maximization of the number of blocks in the goal state, we would not like $P_2^a$ or $P_3^a$ to be considered. This may be done by giving a very high value to *Diff*($Blocks(n_1)$, $Blocks(n_2)$). When only the supervisor's ultimate goals are taken into account, this approach causes the subordinate agent to relate, in its choosing of a plan, to the supervisor's goals, rather than to the naive plan the supervisor generated with partial information. This, however, requires a more complete model of the supervisor than we previously assumed (see Section 7 below).

**(3) Using the domain operators:** Another possibility is to measure the deviation of two states with regard to the operators that generated them. When comparing two states, the metric will be based on how the current state was reached, with particular operators considered as causing greater variations than others. For example, we might define *Diff*($Go(s_i, s_j)$,$Go(s_k, s_l)$) $=$ $\mid s_j - s_l \mid$ to reflect the deviation or convergence caused by following these two operations. This measure is most appropriate for use with the sequential deviation metric ($D_s$). In the example, defining *Diff*($Carry(b, s_i, s_j)$, $Paint(b, c)$) $> 20$ would make $P_1^a$ inferior to all other plans according to all the suggested metrics.

**(4) Considering the plan that unifies the two states:** According to this approach, the metric between any two states is defined by the plan needed to bring about the differing predicates of one state starting from the other: $\mathit{Diff}(s_1, s_2) = \max(P^*(s_1, s_2), P^*(s_2, s_1))$, where $P^*(s_1, s_2) = \min_P P(a, s_1, s_2)$ (the minimization criterion may be the cost of the plan or just the number of operations). A further consideration might also be to take into account the perspective of the supervisor:
$\mathit{Diff}(s_1, s_2) = \max(P^*_a(s_1, s_2), P^*_a(s_2, s_1), P^*_s(s_1, s_2), P^*_s(s_2, s_1))$. Notice that the use of such a metric makes the outcome metric (i.e., bounding KB deviation) redundant.

All of the distance computations above may be costly to perform directly, but the calculation of distance between sets need not use a single distance function uniformly. For example, one might use a (relatively cheap) approximate distance function to map states into an n-dimensional geometric space. Only then would one use the states' approximate "locations" to decide on which ones to apply the more expensive and accurate distance function.

# 6 Additional Considerations

## 6.1 Controlling the Message Flow

We assume that the supervisor is updated solely through the messages sent by the agent. Communication is expensive, especially in the attention it consumes from the supervisor (and too much communication might easily swamp the supervisor with unimportant data). On the other hand, communication is the only way to keep the supervisor in the picture. An important part of the control mechanism is the "reporting policy" that it imposes upon the agent. In order to simplify this problem, we adopt the following assumptions:

- The communication is guaranteed within a fixed time-limit, therefore any message sent by either agent can be considered common knowledge within this time-limit [Halpern and Moses1984].
- There exists a *cost function* ($C_m\colon \mathit{Send}(M) \rightarrow \mathbb{R}$) over the agent's operation of sending messages. The Send operation allows the agent to

send predicates to the supervisor. An obvious function would have the cost directly depend on the length of the message.

There are two situations when the agent should update the supervisor. The first case is whenever the agent discovers a "fantastic" event, typically one that contradicts the agent's basic domain axioms. In that case, we want the agent to report at "any cost." As an example, the Mars-based autonomous vehicle might suddenly discover radioactivity, when its axioms stated that radioactivity does not exist on Mars. Reporting this fact is necessary. The designer of the agent might in fact place specific axioms into the agent's KB to keep the agent on the lookout for their contradiction.

If we had a cost measure for the *influence* that a message has on the supervisor's KB (actually, the cost of the revision process the message triggers in the supervisor), it would seem to be appropriate to *subtract* this influence cost from the cost of the message itself. This is because if the message cost were *increased* to take into account the supervisor's revision process, the more expensive the message would be, the more important it would be that it be sent. By subtracting, we would be doing the opposite: the agent would be encouraged to report to the supervisor any discovery that greatly alters the latter's KB.

The second need for sending a message arises when the agent discovers that its knowledge of the domain prevents him from fulfilling the supervisor's goal, or from following its intended plan (with respect to the policy constraints). In that case, notifying the supervisor about the "stumbling block" may be sufficient to change the supervisor's demands, or to make the supervisor's intended plan come closer to the plans that the agent has in mind. There may actually be many pieces of information that are relevant, and the agent could choose which to report. Thus, this part of the reporting policy is actually a planning problem.

A simple way to incorporate the consideration of reporting cost within the planning process is to add a constraint on the overall cost of messages sent by the agent. This constraint should limit the number of messages in a way that reflects the tradeoff between communication and the other constraints mentioned above. Thus, supplying the agent with another freedom parameter ($T_m$), it would follow that $\sum_k C_m(Send(M_k)) \leq T_m$. The agent can reduce unpredictable consequences via messages, much as the supervisor controls the agent by limiting the latter's unpredicted actions.

Assuming that, in our example, $C_m(Send(\text{“}Blocks(4), Stacked(W, W, s_3)\text{”}))$ $=2$, would make the plan $\langle Send(\text{“}Blocks(4), Stacked(W, W, s_3)\text{”}), Go(s_4, s_3)$, $Carry(W, s_3, s_4)$, $Go(s_4, s_5)$, $Carry(B,s_5,s_3)\rangle\{10\}$ be preferred by all the suggested measurements.

### 6.2 Integration of Private Goals

Beside goals given to it by the supervisor, the agent might have its own goals. Although the agent's function is mainly to achieve the supervisor's goals, it might be useful if there were a way for the agent to fulfill its own goals while pursuing a plan given by the supervisor.

One way to accomplish this is to soften the planning constraint above such that the agent may deviate a certain amount from the intended plan as long as this deviation improves its own utility. For example, the "cost metric" would become $\mid C(P^a) - C(P^m) \mid -u_a(P^a)$ (or, if a bounding parameter is used, $\leq T_c + u_a(P^a)$). Being somewhat more restrictive, the overall utility of the agent might be bounded by some $T_u$.

If the agent's utility of destroying a white block is 5 in the example, then following the new cost metric $P_2^a$ or $P_3^a$ would be chosen.

## 7 Discussion and Future Work

The point of view taken throughout this paper has been that the supervised agent, who has a model of the supervisor and superior information regarding the immediate environment, attempts to choose a plan that doesn't deviate too much from the supervisor's intended plan. The motivation for this has been that the supervisor might be aware of unstated global considerations that are unknown to the supervised agent. The reason that the supervised agent is trying to stick close to the supervisor's plan is thus that the latter may have intended his plan's activity to be carried out, due to implicit constraints or rationales.

There are other ways of looking at the subordinate/supervisor relationship. For example, the subordinate agent might rephrase the question, and ask "What would the supervisor have planned, had he known all the information that I know?" The difference is subtle but important. Instead of comparing realistic plans with the supervisor's naive plan, and taking the closest (as we do above), we might ask how the supervisor's planning itself would have changed given more domain information.

This point of view holds obvious appeal, since clearly the supervisor wants the subordinate to act as he would in the information-rich situation. However, having the subordinate carry out this reasoning process would require a much greater degree of knowledge regarding the supervisor. Above, we suggested that our techniques are suitable when a complete plan, partial plan, or high-level goal has been communicated from the supervisor to the subordinate. In the worst case (where only a goal has been sent), the subordinate is still modeling the supervisor's planning in a specific information context. To consider how the supervisor would use the information in the real world, the subordinate would actually need to know how the supervisor would plan in *any* situation, a global view that cannot reasonably be available to the subordinate. The expectations are too extreme; if the subordinate could second-guess the supervisor in every situation, the subordinate's model of the supervisor would have to be all-encompassing. Our techniques are considerably more modest in aims and scope, but ultimately require more limited and realistic models of the supervisor.

The techniques presented in this paper need to be evaluated in empirical studies. For example, both the inter-plan metrics and the inter-state metrics must be analyzed to determine which are suitable for particular domains. As part of our current research, we are designing and carrying out simulations of subordinate/supervisor planning in various domains, and evaluating how the different approaches outlined above can be used. We are also searching for heuristic methods that will reduce the amount of computation required by the above techniques. For example, an agent might be able to replace the exact Hausdorff metric computations with a heuristic evaluation of inter-plan distance (e.g., instead of taking a representative state from among all states, as the Hausdorff metric requires, we might identify that representative state from a sampling of states).

# 8 Conclusions

We have considered how a subordinate agent might choose among alternate plans for achieving his supervisor's goals. The techniques exploit the knowledge that the subordinate has about the supervisor, and allow the agent to choose its actions appropriately. The discussion focused on metrics for evaluating the distance between plans, considering several techniques (such as the potential cost of moving from one final state to the other, and a cost on difficult to reverse consequences). One key point is that a plan can be considered a set of states, and the distance between plans can be modeled using a distance metric between sets of states (e.g., the Hausdorff metric). To use any distance metric between sets of states, it is also necessary to establish a suitable distance metric between individual states.

Techniques for making plan comparisons can serve in other scenarios where plans are to be compared (for example, maintaining consistency among several subordinate agents, choosing among agents to carry out a plan, predicting the likelihood that a plan will be carried out). Issues such as these will be of importance in building flexible multi-agent systems.

## Acknowledgments

We would like to thank Barbara Grosz, Michael Werman, and several anonymous referees for their comments on previous versions of this paper. This research was partially supported by the Israeli Ministry of Science and Technology (Grant 032-8284).

## References

[Arkin *et al.*1990] Arkin, E.; Chew, L. P.; Huttenlocher, D. P.; Kedem, K.; and Mitchell, J. S. B. 1990. An efficiently computable metric for comparing polygonal shapes. In *Proceedings of the First ACM-SIAM Symposium on Discrete Algorithms.*

[Atallah1983] Atallah, M. J. 1983. A linear time algorithm for the Hausdorff distance between convex polygons. *Information Processing Letters* 17:207–209.

[Chapman1987] Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32(3):333–377.

[Conway *et al.*1990] Conway, Lynn; Volz, Richard A.; and Walker, Michael W. 1990. Teleautonomous systems: Projecting and coordinating intelligent action at a distance. *IEEE Transactions on Robotics and Automation* 6(2):146–158.

[Durfee and Lesser1987] Durfee, E. H. and Lesser, V. R. 1987. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan. 875–883.

[Durfee1988] Durfee, Edmund H. 1988. *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers, Boston.

[Ephrati and Rosenschein1991] Ephrati, Eithan and Rosenschein, Jeffrey S. 1991. Non-absolute control among intelligent agents: Preliminary report. In *Proceedings of the Eighth Israeli Symposium on Artificial Intelligence and Computer Vision*, Ramat Gan, Israel. 155–179.

[Ephrati and Rosenschein1992] Ephrati, Eithan and Rosenschein, Jeffrey S. 1992. Multi-agent planning as search for a consensus that maximizes social welfare. In *Pre-Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Rome, Italy.

[Finger1986] Finger, J. J. 1986. *Exploiting Constraints in Design Synthesis.* Ph.D. Dissertation, Stanford University, Stanford, CA.

[Genesereth *et al.*1986] Genesereth, Michael R.; Ginsberg, Matthew L.; and Rosenschein, Jeffrey S. 1986. Cooperation without communication. In *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, Pennsylvania. 51–57.

[Ginsberg1986] Ginsberg, Matthew L. 1986. Possible worlds planning. In Georgeff, M. and Lansky, A., editors 1986, *Reasoning about Actions and Plans.* Morgan Kaufmann Publishers, Los Altos, California. 213–243.

[Halpern and Moses1984] Halpern, Joseph Y. and Moses, Yoram 1984. Knowledge and common knowledge in a distributed environment. In *Proceedings of the Third ACM conference on Principles of Distributed Computing.* 50–61.

[Holmstorm and Milgrom1991] Holmstorm, B. and Milgrom, P. 1991. Multitask Principal-Agent analyses: Incentive contracts, asset ownership, and job design. *The Journal of Law, Economics and Organization* 7:24–52.

[Katz and Rosenschein1993] Katz, Matthew J. and Rosenschein, Jeffrey S. 1993. Verifying plans for multiple agents. *Journal of Experimental and Theoretical Artificial Intelligence.* To appear.

[Konolige1982] Konolige, Kurt 1982. A first-order formalization of knowledge and action for a multi-agent planning system. *Machine Intelligence* 10.

[Kraus *et al.*1991] Kraus, Sarit; Ephrati, Eithan; and Lehmann, Daniel 1991. Negotiation in a non-cooperative environment. *Journal of Experimental and Theoretical Artificial Intelligence* 3(4):255–282.

[Lesser1987] Lesser, Victor R. 1987. Distributed problem solving. In Shapiro, Stuart C., editor 1987, *Encyclopedia of Artificial Intelligence.* John Wiley and Sons, New York. 245–251.

[Malone *et al.*1988] Malone, Thomas W.; Fikes, Richard E.; and Howard, M. T. 1988. Enterprise: A market-like task scheduler for distributed computing environments. In Huberman, B. A., editor 1988, *The Ecology of Computation.* North-Holland Publishing Company, Amsterdam. 177–205.

[Malone1986] Malone, Thomas W. 1986. Organizing information processing systems: Parallels between human organizations and computer systems. In Zacharai, W.; Robertson, S.; and Black, J., editors 1986, *Cognition, Computation, and Cooperation.* Ablex Publishing Corp., Norwood, NJ.

[Rosenschein1982] Rosenschein, Jeffrey S. 1982. Synchronization of multi-agent plans. In *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania. 115–119.

[Sycara1988] Sycara, Katia P. 1988. Resolving goal conflicts via negotiation. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, St. Paul, Minnesota. 245–250.

[Sycara1989] Sycara, Katia P. 1989. Argumentation: Planning other agents' plans. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan. 517–523.

[Zlotkin and Rosenschein1991] Zlotkin, Gilad and Rosenschein, Jeffrey S. 1991. Incomplete information and deception in multi-agent negotiation. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia. 225–231.