ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-408

Monitoring the Execution of a Multi-Agent Plan: Dealing with Partial Observability

Roberto Micalizio and **Pietro Torasso**[1]

Abstract. The paper addresses the task of monitoring and diagnosing the execution of a multi-agent plan (MAP) which involves actions concurrently executed by a team of cooperating agents. The paper describes a *weak commitment* strategy to deal with cases where observability is only partial and it is not sufficient for inferring the outcome of all the actions executed so far. The paper discusses the role of *target actions* in providing sufficient conditions for inferring the *pending outcomes* in a finite time window. The action outcome provides the basis for computing plan diagnosis and for singling out the goals which will not be achieved because of an action failure.

1 Introduction

The problem of diagnosing the execution of a single-agent plan has been investigated long time ago (see the pioneering work by Birnbaum et al. [1], where the concept of *plan threat* is introduced). However, only recently a number of Model-Based approaches (see [4, 8, 5]) have started to address the complex problem of diagnosing the execution of a multi-agent plan (MAP), i.e. a plan involving a team of cooperating agents, which execute actions concurrently. These works are essentially based on a distributed approach where each agent is responsible for supervising (monitoring and diagnosis) the actions it executes. Typically these approaches assume that action failures are not consequences of plan flaws, but failures are due to the occurrence of unexpected events (such as discrepancies in the shared assumptions or occurrence of faults in some agents functionalities). Thus, the plan execution needs to be supervised in order to detect and explain an action failure as soon as possible. As discussed in [8], the plan diagnosis consists in a subset of actions whose failure is consistent with the anomalous observed behavior of the system. However, this notion of plan diagnosis can be complemented with a notion of threatened actions, which estimates the impact of the failure since the harmful effects of an action failure may propagate to the whole MAP. In this paper, similarly to the previous approaches, a distributed approach for supervising the MAP execution is adopted. However, we address the problem of diagnosing plans characterized by the presence of *joint actions*, which introduce further dependencies among the agents as they need to synchronize and to communicate among themselves. Moreover, we have to deal with actions whose faulty behavior may be non deterministic. In the paper we show that the nominal plan execution imposes some requirement on observability (we will call *minimal observability requirement*) in order to guarantee the inter-agent communication and we introduce a *weak commitment* strategy to deal with cases where observability is only partial and it is not sufficient for inferring the outcome of all the actions executed so far. We will show how the *minimal observability requirement* combined with the *weak commitment* strategy guarantees that the outcome of each action can be inferred within a finite time window.

The paper is organized as follows. In the following sections we introduce the basic notions of global and local plans, then we formalize the processes of monitoring and diagnosis of a MAP and discuss the role of *minimal observability requirement* and *weak commitment* strategy in inferring the actions outcomes which cannot be directly observed; finally we discuss some computational issues and conclude.

[1] Dipartimento di Informatica - Università di Torino, Italy, email: {micalizio, torasso}@di.unito.it

2 Distributed Plan Execution and Supervision

In this paper we consider a specific class of MAS where a team $\mathcal{T}$ of agents cooperate to reach a common complex goal G. In particular, the global goal G is decomposed into a set of (easier) sub-goals, each of which is assigned to an agent in the team. In most cases, however, the sub-goals are not independent of one another as the agents have to cooperate by exchanging services or by executing joint actions; this cooperative behavior introduces causal dependencies among activities, hence when an unexpected event causes the failure of an agent activity, this failure may propagate in the whole system affecting the activities of the other agents in the team.

Global plan. The notion of multi-agent plan (MAP), as formalized by Cox et al. in [2], is well suited for modeling both the agents activities and the causal dependencies existing among them. According to [2], given a team $\mathcal{T}$ of agents, the MAP is the tuple $\langle A, E, CL, CC, NC\rangle$ such that: A is the set of the action instances the agents have to execute; each action a is assigned to a specific agent i of the team $\mathcal{T}$ and it is modeled in terms of preconditions and direct effects. Within the set A there are two special actions: a_0 and a_∞; a_0 is the starting action, it has no preconditions and its effects specify which propositions are true in the initial state; a_∞ is the ending action, it has no effects and its preconditions specify the propositions which must hold in the final state i.e., the preconditions of a_∞ specify the MAP's goal G.

While E is a set of precedence links between actions, CL is a set of causal links of the form $l : a \xrightarrow{q} a'$; the link l states that the action a provides the action a' with the service q, where q is an atom occurring in the preconditions of a'; finally, CC and NC are respectively the *concurrency* and *non-concurrency* symmetric relations over the action instances in A; in particular, the pair $\langle a, a'\rangle$ in CC models a joint action whereas constraints in NC prevent the conflicts for accessing the resources; this is equivalent to the *concurrency requirement* introduced in [8].

Plan Distribution The execution of the MAP P is a critical step as the agents have to concurrently execute the actions assigned to them without violating the constraints introduced in the planning phase. It

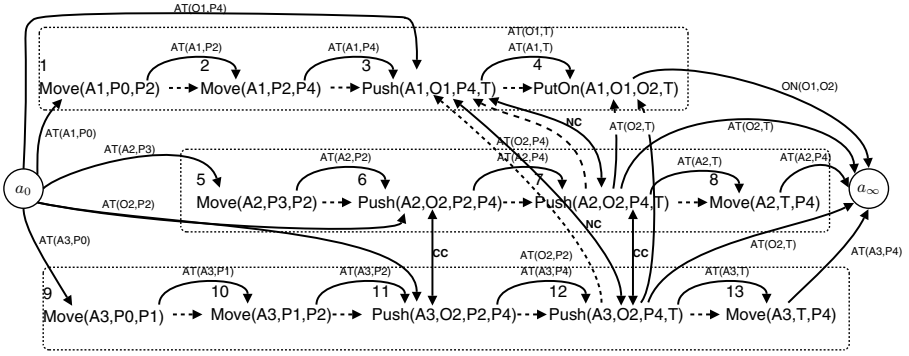

Figure 1. The MAP P to be monitored.

is therefore quite natural to conceive a distributed approach to the supervision of the plan execution. In this paper we adopt a distributed approach to the supervision (similar to the ones discussed in [8, 5]) where each agent performs a (local) control loop over the actions it executes.

Local Plans. The MAP P under consideration is decomposed into as many sub-plans P_i as the agents in $\mathcal{T}$ and each sub-plan P_i is assigned to agent i. The decomposition can be easily done by selecting from P all the actions an agent i has to execute. Formally, the sub-plan for agent i is the tuple P_i=$\langle\ A_i, E_i, CL_i, CC_i, NC_i\ \rangle$ where: A_i, E_i, CL_i, CC_i and NC_i are the same as in P restricted to the actions agent i has to execute (i.e., at least one action belongs to A_i).

We consider the time as a discrete sequence of instants; the actions are executed synchronously by the agents in the team and each action in P takes a time unit to be executed (this common assumption is also made in [8, 6]). At a given time t, an agent i can execute just one action a (in the following the notation a_t^i will denote the action executed by agent i at time t). After the execution of action a_t^i the agent i may receive a set of observations, denoted as obs_{t+1}^i, relevant for the status of i itself.

Minimal Observability Requirement and Agent Communication. Since the agents need to *communicate* for achieving coordination during the plan execution, a *minimal observability requirement* must be satisfied.

To figure out what events are included in the minimal observability requirement consider that coordination is required in three cases during the nominal plan execution. First, when an agent i has to provide a service q to another agent j; technically, this case is encoded by a causal link $l : a \xrightarrow{q} a'$ in the MAP P (where $a \in A_i$ and $a' \in A_j$). After the execution of a, the agent i must be able to observe the achievement (or the absence) of service q and must notify agent j whether the service has been provided or not; in fact, because of partial observability, the agent j can not directly observe the service q and has to wait for a message from i.

The second situation which requires explicit coordination during the execution regards the joint actions. Every pair of actions $\langle a, a'\rangle$, included in the set of concurrency constraints CC, models a joint action; where a and a' are actions assigned to agents i and j, respectively. In order to execute the joint action $\langle a, a'\rangle$ in a synchronized way, the two agents i and j need to observe whether the preconditions of the actions a and a' are satisfied; in fact the joint action can be performed only when the preconditions of both actions are satisfied and both agents have to be aware of this.

Explicit coordination is also required for executing actions bounded by non-concurrency constraints: in this case coordination is ruled by the set of non-concurrency constraints NC in P and prevents the simultaneous execution of the constrained actions. Given the pair of actions $\langle a, a'\rangle \in NC$ (where $a \in A_i$ and $a' \in A_j$ respectively), when agent i intends to execute action a must inform agent j and in case of conflict the two agents have to negotiate.

As we will see in section 4, agents communicate even in case of action failures: an agent must notify other agents when a service will not be provided as a consequence of a failure.

Running Example. In the paper we will use a simple example from the blocks world for illustrating the concepts and the techniques we propose. Let us consider three agents that cooperate to achieve a global goal G where two blocks `O1` and `O2` are moved in a target position `T` and `O1` is put on the top of block `O2`; initially, the blocks are located in position `P4`. In its nominal behavior an agent can move a block by pushing it; however, in some cases a block may be too heavy and two agents need to join their efforts to push it. Figure 1 shows a possible MAP which achieves the goal G, in particular, the agents `A2` and `A3` cooperate to move the (heavy) block `O2` in position `T` (see the joint actions $\langle$`6,11`$\rangle$, $\langle$`7,12`$\rangle$); the agent `A1` move the block `O1` in position `T` than it puts `O1` on the top of `O2` (see action `PutOn`).

The MAP is a DAG where nodes are actions, solid and dashed arrows are causal and precedence links respectively, while concurrent and non-concurrent constraints are solid, bidirectional arrows labeled as CC and NC respectively. The dashed rectangles specify which actions are included in the sub-plans assigned to the three agents.

The operations within the target position are constrained: at each time instant only one block can be moved in, so there are non-concurrency constraints between the joint action $\langle$`7,12`$\rangle$ and the simple action `3`; moreover, since the block `O2` must be positioned in `T` earlier than `O1`, precedence links exist between the actions $\langle$`7,12`$\rangle$ and `3`.

3 Monitoring with uncertain action outcomes

The monitoring performed by agent i over the execution of its sub-plan provides two important services:

1) Estimate the state of agent i after the execution of an action a
2) Detect the outcome of the action a

However, before describing the monitoring process we need to introduce some important concepts.

Agent state. Intuitively, the system status can be expressed in terms of the status variables of the agents in the team $\mathcal{T}$ and of the status of the system resources RES. However, the distributed approach to the supervision prevents the adoption of a global notion of status while it allows a local view based on a single agent.

The status of agent i is expressed in terms of a set of status variables VAR^i, which is partitioned into three subsets END^i, ENV^i and HLT^i. END^i and ENV^i denote the set of endogenous (e.g., the agent's *position*) and environment (e.g., the resources state) status variables, respectively. Note that, because of the partitioning, each

agent i has to maintain a private copy of the resource status variables; more precisely, for each resource $res_k \in RES$ ($k : 1..|RES|$) the private variable $res_{k,i}$ is included in the set ENV^i.

Since we are interested in monitoring the plan execution even when action failures occur, we introduce a further set of variables in order to model the agent faults, which may cause action failures. HLT^i denotes the set of variables concerning the health status of an agent functionalities (e.g., mobility and power); in particular, for each agent functionality f, a variable $v_f \in HLT^i$ represents the health status of f, the domain of variable v_f is the set $\{ok, abn_1, \ldots, abn_n\}$ where ok denotes the nominal mode while $abn_1, \ldots, abn_n$ denote non nominal modes.

It is worth noting that the observations obs_t^i agent i may receive, convey information about just a subset of variables in VAR^i. First of all, an agent can directly observe just the status of the resources it is actively exploiting: the status of other resources is not directly observable but an agent can communicate with other agents to determine it. Moreover, the observations obs_t^i provide in general the value of just a subset of variables in END^i; whereas the variables in HLT^i are not directly observable and their actual value can be just inferred. Given this partial observability, at each time t the agent i can just estimate a *set* of alternative states which are consistent with the received observations obs_t^i; in literature this set is known as *belief state* and will be denoted as $\mathcal{B}_t^i$.

Action models. The model of a *simple action* a_t^i (assigned to agent i at time t) is the tuple $\langle var(a_t^i), pre(a_t^i), eff(a_t^i), event(a_t^i), \Delta(a_t^i)\rangle$; where $var(a_t^i) \subseteq VAR^i$ is the subset of status variables over which the set $pre(a_t^i)$ of *preconditions* and the set $eff(a_t^i)$ of *effects* are defined; $event(a_t^i)$ is the set of exogenous events (e.g., faults) which may occur during the execution of action a_t^i and which possibly may affect its outcome; finally, $\Delta(a_t^i)$ is a transition relation where every tuple $d \in \Delta(a_t^i)$ models a possible state transition, which may occur while i is executing a_t^i. Each tuple d has the form $d = \langle s_t, event, s_{t+1}\rangle$; where s_t and s_{t+1} represent two agent states at time t and $t+1$ respectively (each state is a complete assignment of values to the status variables in $var(a_t^i)$) and $event$ (possibly empty) represents the occurrence of an unexpected event in $event(a_t^i)$.

Since $\Delta(a_t^i)$ is a relation, the action model can represent non deterministic, anomalous action effects. The healthy formula $healthy(a_t^i)$ of action a_t^i is computed by restricting each variable $v_f \in healthVar(a_t^i)$ to the nominal behavioral mode ok and represents the nominal health status of agent i required to successfully complete the action itself. Therefore, the (expected) nominal effects of a_t^i are $nominalEff(a_t^i)$=$\{q \in eff(a_t^i) \mid pre(a_t^i) \cup healthy(a_t^i) \vdash q\}$.

On the contrary, when the healthy formula does not hold, the behavior of the action may be non deterministic and some (even all) of the expected effects may be missing.

Joint actions. The notion of simple action can be extended to cover the notion of joint action, which as discussed in [3], can be seen as the simultaneous execution of a subset of simple actions. In this paper we consider a stronger notion of joint action: two simple actions a_t^i and a_t^j are part of a joint action $a_t^{i,j}$ not only because they are executed at the same time, but also because the agents i and j actively cooperate to reach an effect. The notion of *dependency set*, introduced in [5], is exploited to homogenously represent both simple and joint actions. Intuitively, a dependency set $I(t)$ highlights the subset of agents whose strict cooperation is required in a specific time instant t and can be easily determined from the concurrency constraints defined in the MAP. The agents within the same dependency set $I(t)$ synchronize in order to build a joint belief state $\mathcal{B}_t^I$ (resulting from the conjunction of their local belief states) and the joint model of the action $a_t^{I(t)}$ (see details in [5]). Thus, given the dependency set $I(t)$, $a_t^{I(t)}$ may denote a simple or a joint action.

The prediction process. In [5] Micalizio et al. have proposed a distributed strategy for monitoring the execution of a MAP. Their approach can be summarized as follows: let $a_t^{I(t)}$ denote the (joint) action executed by the agent(s) in the dependency set $I(t)$ at time t (for the sake of readability we will write a_t^I whenever the time of the dependency set is obvious from the context), let $\mathcal{B}_t^I$ be the (joint) belief state of the agents in I, let $\Delta(a_t^I)$ the model of the (joint) action the agents in I have to execute at time t, the joint belief state at time $t+1$ (i.e., after the action execution) can be inferred as:

$$\mathcal{B}_{t+1}^I = \pi_{VAR_{t+1}^I}(\sigma_{obs_{t+1}^I}(\mathcal{B}_t^I \bowtie \Delta(a_t^I))).$$

The join operation $\mathcal{B}_t^I \bowtie \Delta(a_t^I)$ represents the prediction step as it estimates the set of possible states of the agents in I at time $t+1$. However, the set of predictions resulting from the join operation is in general spurious as it predicts all possible evolutions. The selection operation $\sigma_{obs_{t+1}^I}$ has the effect of pruning off all those predictions which are inconsistent with the observations received by the agents in I at time t+1 where $obs_{t+1}^I = \bigcup_{i \in I} obs_{t+1}^i$. Of course, the precision of the estimated joint belief strongly depends on the amount of available observations: in the worst case obs_{t+1}^I is empty and the selection operator can not discard any of the predicted states; in the best (unrealistic) case obs_{t+1}^I is so complete to reduce the estimated states to the actual agent state. Finally, the (joint) belief state $\mathcal{B}_{t+1}^I$ is inferred by projecting the set of refined predictions over the agent status variables at time t+1.

Strong Commitment. Intuitively, the outcome of action a_t^I is *succeeded* when all the nominal, expected effects are achieved after its execution, the action outcome is *failed* otherwise. However, since the belief state $\mathcal{B}_{t+1}^I$ may be highly ambiguous, in [5] the authors adopt a strong commitment policy by considering a_t^I as successfully completed iff all its nominal effects $nominalEff(a_t^I)$ have been achieved in every possible state s included in $\mathcal{B}_{t+1}^I$, formally:

$$a_t^I succeeded \leftrightarrow \forall q \in nominalEff(a_t^I), \forall s \in \mathcal{B}_{t+1}^I, s \models q \quad (1)$$

Moreover, [5] requires that the outcome of action a_t^I must be immediately assessed after the execution of the action at time t+1. Therefore, when $nominalEff(a_t^I)$ are not satisfied in at least one state included in the joint belief $\mathcal{B}_{t+1}^I$, the outcome of action a_t^I is assumed to be *failed*. This strong commitment policy is based on the assumption that, whenever the action a_t^I is successfully completed, the amount of observations available at time t+1 is sufficient for pruning off from $\mathcal{B}_{t+1}^I$ any state s where the nominal effects do not hold. Under this assumption it is sufficient that each agent maintains just the last belief state $\mathcal{B}_{t+1}^I$, as it represents a synthesis of the past history till time t+1.

Unfortunately this assumption may not hold in many domains and, as a consequence of the partial observability, it may happen that even when an action is successfully completed an agent concludes a failure because it can not univocally assert a success.

Weak Commitment. In order to avoid this problem we propose a more flexible strategy where the outcome of an action a_t^I can be inferred within a time window rather than at the precise time instant t+1. In particular we assume that the system observability satisfies just the minimal observability requirement, and we propose a methodology for monitoring the plan execution which is able to cope with this constraint. This means that, when an agent is unable to determine the outcome of action a_t^I, the agent does not conclude the failure of a_t^I, but postpones the assessment of the action outcome. In fact, although the outcome of a_t^I can not be precisely determined

at the current time t+1, it may be determined in a future time instant by exploiting observations that each agent i in I will receive. For this reason, each agent i has to maintain a list $pO^i(t)$ of actions whose outcome has not been determined yet at time t; i.e., a list of *pending outcomes*. Moreover, the agent i has to maintain a trajectory $Tr_i[0, t+1]$, which relates all the belief states agent i has inferred so far. In particular, since the belief states are ambiguous and, in general, include a number of alternative states, $Tr_i[0, t+1]$ is a *set of trajectories*.

Refining the agent trajectory. Given the action a_t^I, such that the agent $i \in I$, the process for estimating the belief state of agent i at time t+1 consists in the process for extending the agent trajectory $Tr_i[0..t]$ to cover the time instant t+1. Also in this case we adopt the Relational Algebra operators to formalize this process:

$$Tr_i[0, t+1] = \sigma_{obs^I_{t+1}}(Tr_i[0, t] \bowtie \Delta(a_t^I))$$

The join operator represents the step which extends the agent trajectory, in fact any of the transitions modeled in $\Delta(a_t^I)$ are appended at the end of one (or more) traces in $Tr_i[0..t]$. Observe that the join operator implicitly refines also the agent trajectory: all the traces in $Tr_i[0..t]$ which do not participate to the join are discarded.

The selection operator further refines the agent trajectory as it filters out all those traces which are inconsistent with the observations available at time t+1.

Therefore, $Tr_i[0..t+1]$ maintains all the possible agent trajectories which are consistent with the observations received so far, given a sequence of actions executed by agent i in the interval $[0..t+1]$.

Inferring action outcome. Since the extension of the agent trajectory refines the trajectory itself, it may reduce the ambiguity in some of the previous belief states. Thus, agent i can try to infer the outcome of some of the actions in $pO^i(t+1)$; in fact, for each action $a_k^{I(k)} \in pO^i(t+1)$ (where $I(k)$ represents the dependency set including i at time $k \in [0, t+1]$), it is possible to determine the belief state inferred by the agent at time $k+1$ from the agent trajectory $Tr_i[0, t+1]$ as follows:

$$\mathcal{B}^I_{k+1} = \pi_{k+1}(Tr_i[0, t+1]).$$

Observe that $\mathcal{B}^I_{k+1}$ is potentially different from the belief state inferred by the agent i at time k, in fact $\mathcal{B}^I_{k+1}$ results from the progressive extension of the agent trajectory from time k to time t+1 and at each step $\mathcal{B}^I_{k+1}$ may have been refined. The nominal outcome of action a_k^I is therefore inferred similarly to the definition in formula 1; i.e., if the nominal effects of the action a_k^I hold in every state $s \in \mathcal{B}^I_{k+1}$ the action outcome is *succeeded*. However, the achievement of the nominal effects of action a_k^I is a consequence not only of the nominal execution of this action but also of the previous actions which are causally related to a_k^I. The relation between the nominal outcome of a_k^I and the previous actions is formalized in the following property:

Property 1 *Given the agent i and its dependency set I at time k, let a_k^I be an action with outcome* succeeded*, then all the actions a_h in $pO^i(k) \cap dependsOn(a_k^I)$ have outcome* succeeded *too*

where $dependsOn(a_k^I)$ denotes the subset of actions $\{a_1, \ldots, a_n\}$ in A_i, which directly or indirectly provide a_k^I with a service (i.e., through a sequence of causal links).

It is also possible that the extension of the trajectory does not sufficiently refine the belief state $\mathcal{B}^I_{k+1}$ in such a way the nominal effects of the action a_k^I hold just in a subset of the states included in $\mathcal{B}^I_{k+1}$; in this case the outcome of a_k^I remains pending.

Finally, if the nominal effects do not hold in any state included in $\mathcal{B}^I_{k+1}$ we conclude that the outcome of a_k^I is non nominal. This does not necessarily imply that the action a_k^I is failed since the not achievement of the nominal effects may depend on the failure of previous actions causally related to a_k^I.

4 Plan Diagnosis.

As soon as the outcome of an action a_t^I is determined to be non nominal , a diagnostic process is activated in order to provide a possible explanation for such a non nominal outcome. In this paper we adopt the same notion of *plan diagnosis introduced* by Roos et al. in [8]: once observed a non nominal outcome of action a_t^I, the plan diagnosis $PD(a_t^I)$ singles out a subset of actions executed by the agents in I, whose failure is consistent with the anomalous, observed behavior of the system.

Given an agent $i \in I$, every action a in $EXP^i(a_t^I) = (pO^i(t) \cap dependsOn(a_t^I)) \cup a_t^I$ is a minimal explanation of the non nominal outcome of a_t^I. Therefore, the plan diagnosis for agent i is $PD^i(a_t^I) = \bigvee_{a \in EXP^i(a_t^I)} a$; in fact, due to the causal dependencies, it is sufficient to assume the failure of at least one of these actions to explain the observed non nominal outcome of a_t^I. It is east extending the plan diagnosis to the dependency set I as $PD(a_t^I) = \bigvee_{i \in I} PD^i(a_t^I)$.

Essentially, the plan diagnosis explains the observed, *non nominal outcome* of a_t^I by singling out a subset of actions whose failure may be the root cause of that observation.

Missing Goals. The plan diagnosis can be refined by determining the set of *missing goals*. A missing goal is a service which can not be provided by agent i as a consequence of the failure of action a_t^I (where i belongs to the dependency set I). To formally characterize the concept of missing goal we introduce the notion of *primary effect*: given an action a^I the nominal effect q in *nominalEff*(a^I) is a primary effect if at least one of the following conditions hold:

1. $q \in pre(a_\infty)$ i.e., q belongs to the global goal.
2. q is a service that a^I provides to a subset J of agents; i.e., there exists a causal link $l : a^I \xrightarrow{q} a^J$ where $I \neq J$. Observe that a^I and a^J can be joint or simple actions.

In general, given an action a^I, *primary*(a^I) denotes the (possibly empty) set of primary effects provided by a^I.

To determine the set of missing goals we adopt a conservative policy and we assume that all actions included in plan diagnosis are actually failed. Therefore, the subset of missing goals that the agents in I can no longer achieve is: *missingGoals*$(a_t^I) = \bigcup_{a \in PD(a_t^I)}$*primary*$(a)$.

In principle, it is sufficient to achieve all the missing goals in an alternative way in order to reach the MAP's global goal G despite the occurrence of the failure. Therefore the missing goals may be the starting point for any plan recovery strategy.

Propagating the Plan diagnosis. As said above, the failure of a_t^I may propagate in the plan preventing the execution of actions assigned to different agents in the team (not limited to the dependency set I), possibly causing the stop of the whole system. For this reason, we complement the notion of plan diagnosis with the set *threatened actions ThrActs*(a_t^I) which could be indirectly affected by the failure of a_t^I (through a sequence of causal links). Intuitively, an action is threatened through a causal link $l : a' \xrightarrow{q} a$ when it is no longer guaranteed that the action a' provides the service q; this may happen either because a' is failed (i.e., included in the plan diagnosis $PD(a_t^I)$) or because a' is in turn threatened. Formally the set *ThrActs*(a_t^I) is defined as:

$$\textit{ThrActs}(a_t^I) = \{a \in A \mid a_t^I \prec a \text{ and } \exists \text{ a causal link } l : a' \xrightarrow{q} a, \; l \in CL, a' \in PD(a_t^I) \text{ or } a' \in \textit{ThrActs}\}$$

Observe that, the propagation is a form of communication among agents, which conveys negative information; hence an agent does not wait indefinitely for services which will never be provided.

Running Example. Let us consider the blocks world example and assume that at time 4 the failure of the joint action $\langle 7, 12 \rangle$

(whose dependency set is {A2,A3}) is detected. To determine whether this failure may be consequence of a previous failure, one has to single out which actions in $pO^{A2}(4)$ ($pO^{A3}(4)$) directly or indirectly provide $\langle 7,12\rangle$ with a service. According to the definition of primary service, the outcome of actions 5 and 10 must be observable, whereas the outcome of action $\langle 6,11\rangle$ may be not. Let us suppose that the observations available at time 3 are not sufficient for inferring the outcome of action $\langle 6,11\rangle$; thereby both the agents A2 and A3 include the joint action $\langle 6,11\rangle$ in the set pending outcomes; i.e., $pO^{A2}(4)$=$pO^{A3}(4)$={$\langle 6,11\rangle$}. Since $\langle 6,11\rangle$ directly provides a service to action $\langle 7,12\rangle$, the failure of the second action may be a consequence of the failure of the first one, thus the plan diagnosis includes both actions: $PD(\langle 7,12\rangle)$={$\langle 6,11\rangle, \langle 7,12\rangle$}. Given the plan diagnosis, the set of missing goals is *missingGoals*($\langle 7,12\rangle$)={AT(O2,T)}; moreover the propagation of the plan diagnosis highlights that the failure of action $\langle 7,12\rangle$ affects not only the actions 8 and 13 of the agents A2 and A3 respectively, but also the action 4 of the agent A1. Note that providing the missing service AT(O2,T) the agent A1 would be able to accomplish its task without any adjustment to its sub-plan.

5 Computational Issues

So far we have discussed in a declarative way a methodology for supervising a MAP, in this section we analyze some computational issues which may arise while implementing the approach.

Agents Trajectories. Since maintaining a set of trajectories from the initial time instant may be computationally expensive, we can limit the length of the agent trajectories by considering that the primary effects of an action a_t^I must always be observable at time t+1 (according to the minimal observability requirement). In order to make evident in the MAP which actions provide primary effects we introduce the notion of *target action*; in particular, an action a_t^I is said to be a target action iff *primary*(a_t^I) is not empty. Since the outcome of a target action is always observable, target actions can be considered as milestones in the plan and exploited for determining temporal windows the agent trajectories must cover. In particular, under some requirements on the causal dependencies in the MAP, the following property holds.

Property 2 *Given the agent i, the target action a_t^I, where $i \in I$, and the set $pO^i(t)$, if each action in $pO^i(t)$ provides (directly or indirectly) a_t^I with a service, the detection of the outcome of a_t^I allows to infer the outcome of each action included in $pO^i(t)$.*

Property 2 states that, after the execution of a target action a_t^I, every agent $i \in I$ can determine the outcome of all the actions in the set of pending outcomes $pO^i(t)$. Moreover, in case no failure has been detected, every agent i can replace the trajectory with the belief state $\mathcal{B}_{t+1}^I$ as it represents a synthesis of the past history till time t+1.

For example, in the MAP of Figure 1, the simple actions 4,5,8,13 are target actions as well as the joint action $\langle 7,12\rangle$.

Implementation and preliminary results From a computational point of view, managing relations such as belief states and action models which possibly have a huge dimension may be very expensive. In order to implement both the monitoring and diagnostic processes in an effective way, we have encoded the relation by means of the symbolic formalism of the Ordered Binary Decision Diagrams (OBDDs); the relational operations have been mapped into standard operations on OBDDs.

A prototype has been implemented in Java JDK 1.6 and exploits the JavaBDD(http://sourceforge.net/projects/javabdd) package for manipulating OBDDs. The approach has been tested in a office domain and the robotic agents, simulated in a software environment, are implemented as threads running on the same Intel Pentium (1.86 GHz, RAM 1 GB, Windows XP OS). The preliminary results collected so far are encouraging: given MAPs involving up to 6 agents and 60 actions on average, the plan supervision (monitoring, agent diagnosis and failure propagation) performed by each agent requires on average 5 msec. per instant (being the maximum absolute CPU time per instant 30 msec); exploiting the target actions, an agent maintains a trajectory whose length is 5 instants in the worst case (3 on average).

6 Discussion and Conclusion

The problem of diagnosing a multiagent plan has been recently addressed by exploiting methods and techniques developed within the MBD community, in particular for the diagnosis of distributed system (see e.g., [7]). In [4] the authors consider multi-agent systems where, at each time instant, every agent chooses the more appropriate behavior to assume according to its beliefs. The authors introduce the notion of *social diagnosis* to explain the disagreements among cooperating agents.

The approach presented in this paper has some resemblance with [8], where a distributed approach to monitoring and diagnosing the execution of a MAP is proposed. It assumes that each agent monitors and diagnoses the actions it is responsible for where actions are atomic and are modeled as functions of their nominal behavior only. Since the anomalous behavior of the actions is not explicitly modeled, the monitoring can not estimate faulty system states.

In this paper, we propose a distributed approach for monitoring and diagnosing the execution of a multi-agent plan in a system which is only partially observable. Differently from the approach in [8], we adopt extended action models for capturing both nominal and anomalous execution. Thereby the monitoring process we propose is able to estimate system states even after the occurrence of faults. Moreover, by exploiting the notion of dependency set introduced in [5], the approach uniformly deals with simple as well as joint actions.

Finally the paper has discussed a methodology based on the weak commitment strategy which is able to infer a plan diagnosis and to determine two important pieces of knowledge over the system status: the set of missing goals and the set of actions threatened by the plan diagnosis. These two sets play a critical role in any plan recovery strategy since one has to find (if possible) an alternative way for reaching the global goal which is not achievable because of the action failure. In general such a recovery step requires a re-planning phase where the set of missing goals contribute to reduce the search space since they clearly point out what must be achieved.

REFERENCES

[1] L. Birnbaum, G. Collins, M. Freed, and B. Krulwich, 'Model-based diagnosis of planning failures', in *Proc. AAAI'90*, pp. 318–323, (1990).
[2] J. S. Cox, E. H. Durfee, and T. Bartold, 'A distributed framework for solving the multiagent plan coordination problem', in *Proc. AAMAS05*, pp. 821–827, (2005).
[3] R. M. Jensen and M. M. Veloso, 'Obdd-based universal planning for synchronized agents in non-deterministic domains', *JAIR*, **13**, 189–226, (2000).
[4] M. Kalech and G.A. Kaminka, 'Towards model-based diagnosis of coordination failures', in *Proc. AAAI05*, pp. 102–107, (2005).
[5] R. Micalizio and P. Torasso, 'On-line monitoring of plan execution: a distributed approach', *Knowledge-Based Systems*, **20**(2), 134–142, (2007).
[6] Roberto Micalizio and Pietro Torasso, 'Plan diagnosis and agent diagnosis in multi-agent systems', volume 4733 of *LNCS*, pp. 434–446, (2007).
[7] Y. Pencolé and M.O. Cordier, 'A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks', *AI*, **164**, 121–170, (2005).
[8] C. Witteveen, N. Roos, R. van der Krogt, and M. de Weerdt, 'Diagnosis of single and multi-agent plans', in *Proc. AAMAS05*, pp. 805–812, (2005).