Diagnosis of Multi-Agent Plans under Partial Observability

Roberto Micalizio **Pietro Torasso**
Università di Torino, Dipartimento di Informatica
corso Svizzera 187, Torino, Italy; tel. (+39) 011 6706773 - fax. (+39) 011 751603
{micalizio,torasso}@di.unito.it

Abstract

The paper discusses a distributed approach for monitoring and diagnosing the execution of a plan where concurrent actions are performed by a team of cooperating agents.

The notions of plan diagnosis and agent diagnosis are introduced to model primary and secondary failures and the root causes of the primary failures respectively. The paper addresses the problem of diagnosing multi-agent plans when the partial observability of the status of the system does not allow to univocally determine the outcome of all the actions executed so far. In particular, we presents a methodology where the detection of the outcome of *target actions* activates a diagnostic process, aimed at estimating the outcome of other actions previously executed and at detecting, if any, primary failures. Such diagnostic inferences are then propagated to the other agents for determining the secondary failures.

In the monitoring and diagnostic steps, the system makes use of a relational model of the actions for capturing both the nominal and the abnormal result of their execution.

Introduction

In the AI community it is widely recognized that multi-agent systems constitute a powerful approach for solving complex tasks in a distributed way. Distributing a task among cooperating agents of a team offers several advantages: the agents can pursue different goals in parallel, distributed resources may be used more efficiently, and multi-agent systems are more robust since the risk of a global failure is reduced.

However, distributed problem solving presents a number of challenges to deal with (see e.g. (Carver & Lesser 2003) in particular, the execution of a multi agent plan (MAP) can be *threatened* (Birnbaum *et al.* 1990) by the occurrence of unexpected events, such as faults in the agents or harmful interactions which arise when a number of agents try to use the same resources simultaneously. Although in some particular cases it is reasonable to assume that the planner is able to prevent all kinds of threats (Boutilier & Brafman 2001), this is in general unfeasible (think for example to the need of anticipating all the possible occurrences of faults that may affect the functionalities of the agents).

The occurrence of plan threats does not necessarily imply that the plan goal can no longer be achieved. In many cases, in fact, the plan goal can still be achieved but the plan may need to be adjusted (i.e. a recovery process is required). In order to repair a multi-agent plan, one has to know not only which actions are failed (*primary failures*), but also how the rest of the plan has been affected by these failures (*secondary failures*). In other words, the execution of a multi-agent plan needs to be *on-line monitored* and *diagnosed* to take into account possible anomalous evolutions during the execution of the plan.

It is worth noting that a MAP is substantially different from a component-based system. The behavior of a component-based system essentially depends on the health status of its components and on the commands submitted to them. The relations among the system components (e.g., the topological architecture of the system) usually do not change over time. Therefore the model of a component-based system can be defined a-priori and does not change during the diagnosis.

On the contrary, the behavior of a MAP depends both on the actions the agents execute and on the occurrence of faults in the functionalities of these agents. As a consequence, the relations among the agents (e.g., competition or cooperation) change over time and the model of the MAP needs to be adjusted according to the current set of actions.

Moreover, also the notion of diagnosis of a MAP is different from the notion of diagnosis of a component-based system. In fact, the classical notion of model-based diagnosis of a component-based system consists of a (minimal) set of system components which are assumed faulty to explain the anomalous observed behavior of the system itself. On the contrary, as discussed in (Witteveen *et al.* 2005), the notion of diagnosis of a MAP consists of a (minimal) subset of actions executed by the agents, whose failure explains the observed behavior.

Finally, one of the peculiar characteristics of a MAP is represented by the fact that the agents of the team typically adopt a cooperative behavior. In general, the cooperation consists of an agent i that provides another agent j with some services. In these cases, a fault in the functionalities of agent i, causing the failure of the current action of i (*primary failures*), may cause the failure of some actions assigned to j (*primary failures*), which in turn may have cascade effects on the actions assigned to other agents in the team.

After this premises it is evident that the model-based methodologies developed to monitoring and diagnosis a

component-based system are just partially applicable when the system under consideration is a MAP. In this paper we discuss and formalize a novel model-based methodology to solve the problems of monitoring and diagnosing the execution of a MAP with atomic actions. In particular we propose a distributed methodology, where each agent is responsible for monitoring and diagnosing the sub-plan it executes. We assume that actions are atomic and we model them as relations, which capture their effects both in the nominal and in the anomalous situations.

As one of the main contributes of this work, we provide two notions of diagnosis: the notion of *plan diagnosis* highlights which primary and secondary failures have occurred, while the notion of *agent diagnosis* explains every primary failure in terms of a combination of faults occurred in the functionalities of an agent; moreover the paper points out the relation between the agent and the plan diagnosis.

As a further contribution, the methodology formalized in this paper is able to deal with *weakly observable systems*; under the conditions of weak observability an agent may be unable to determine the outcome of every action it executes. This means that an agent must be able to exploit the observations available at a given time to determine, if possible, the outcome of some actions previously executed.

The paper is organized as follows. In the following section we introduce the basic notions of global and local plans, then we formalize the processes of monitoring and diagnosis of a MAP. The paper closes with some concluding remarks.

Setting the Framework

The class of systems we deal with is referred to as *synchronous transition systems* (Kurien & Nayak 2000). In these systems the time is 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 is a common assumption, see e. g. (Witteveen *et al.* 2005)); finally, at each instant some observations are available (typically the system is just partially observable).

The problem of supervising (monitoring + diagnosing) the execution of P is decomposed into a set of sub-problems: the global plan P is decomposed in as many sub-plans as the agents in the team $\mathcal{T}$; each sub-plan P_i is assigned to agent i, which is responsible for executing and supervising the actions in P_i. At each time instant t, an agent i receives a set of observations obs_t^i relevant for the status of agent i itself, although in general the observations obs_t^i are not sufficient for precisely inferring the status of agent i.

The partitioning of activities among agents described above does not guarantee that the sub-problems are completely independent of one another. In fact, we deal with the case where the agents in $\mathcal{T}$ cooperate to achieve a common global goal G. The cooperation among agents consists in the exchange of services, as will be formalized in the following of this section. In particular, given an agent i and its next action a, a is *executable* iff the set of preconditions *pre*(a) are satisfied in the current status estimated by i; some of the preconditions in *pre*(a) may be services, which other agents in $\mathcal{T}$ have to provide; that is, there exists a causal link $l : a' \xrightarrow{q} a$, where $a' \in A_j$. When agent i intends to execute action a, the service q may be not provided yet, in this case we assume that agent i voluntarily waits for the satisfaction of q. In other words, agent i slightly adjusts its sub-plan P_i by adding a *Wait* action. On the other hand, when agent j is able to determine whether the service q has been achieved or not (i.e., agent j determines the outcome of action a'), it notifies the waiting agent i about the status of the service q; in this way we avoid that agent i waits indefinitely for the required services. To keep the discussion simple, in this paper the outcome of an action can be either *succeeded*, when the action is executed in the nominal way, or *failed* otherwise.

It is worth noting that the cooperative behavior among agents introduces causal dependencies among the actions the agents have to execute; therefore, when an action failure occurs in the sub-plan P_i (i.e., a *primary failure* occurs), the failure may affect the sub-plans of other agents, namely, the failure may propagate not only in P_i, but even in the global plan P. According to the traditional terminology (see e.g., (Pencolé & Cordier 2005)), the actions which fail as a direct or indirect consequence of a primary failure are called *secondary failures*.

Albeit the supervision of the execution of plan P is decomposed into a set of sub-problems, these sub-problems can not be solved independently of one another. In fact, the agents need to communicate during the supervision process, in order to determine, in a distributed way, a plan diagnosis which highlights the distinction between primary and secondary failures. Moreover, we formalize the relation between primary failures and agent diagnoses and show the critical role played by the agent diagnosis to determine the set of primary failures.

The structure of a Multi-Agent Plan. In this section we briefly introduce some basic notions on the multi-agent plan. In particular, we assume that the global plan P to be monitored is synthesized by means of a planner similar to the partial-order planner (POMP) by Boutilier and Brafman (Boutilier & Brafman 2001). The POMP planner considers atomic, STRIPS-like actions (i.e. an action a is modeled in terms of its preconditions *pre*(a) and effects *effects*(a)); the STRIPS language is augmented with a *concurrent* clause which specifies, for each action a, a list of actions of other agents which can, or can not, be executed concurrently with a. During the planning process, the POMP keeps track of two kinds of links between actions:

- *causal* links, e.g. $a \xrightarrow{q} a'$ indicates that the action a provides the action a' with the service q, where q is an atom occurring in the preconditions of a';

- *precedence* links, e.g. $a \prec a'$ indicates that the action a must precede the execution of the action a'. These links are used to resolve agents competitions: when two (or more) agents need to access the same resource, the POMP produces a plan where the access to the resource is serialized.

We require that the global plan P satisfies the following characteristics:

- the actions assigned to an agent are totally ordered

- the precedence links among actions of different agents are translated into causal links where the offered service consists in relinquishing a resource. For example, the precedence link $a \prec a'$ rules the access to the resource *res* be-

tween the action a, assigned to agent i, and a', assigned to agent j. This precedence link is replaced by the causal link $a \stackrel{free(res)}{\longrightarrow} a'$ to make clear that role played by the service *free(res)*which enables action a' since *free(res)* appears in the preconditions of a'.

Formally, the global plan P is a tuple $\langle A, <, C\rangle$, where A is the set of actions the agents have to execute, $<$ is an order relation between actions assigned to the same agent, C is a set of causal links of the form $a \stackrel{q}{\rightarrow} a'$ where a and a' may or may not be assigned to the same agent.

Main services and target actions Given an agent $i \in \mathcal{T}$ and an action a assigned to i (i.e., $a \in A_i$), *main*(a) denotes the set of main services produced by a, more precisely each effect $q \in$*main*(a) satisfies at least one of the following conditions:

- $q \in G$ i.e., q is an atom which appears in the goal G of the global plan.

- $q \neq$ *free(res)* and it is a service agent i provides to another agent j, that is, there exists a causal link $a \stackrel{q}{\rightarrow} a'$ where $a' \in A_j$.

An action $a \in A_i$ such that *main*$(a) \neq \emptyset$ is said a *target action*, the action is said *simple* otherwise. Given an action a, either target or simple, *depends_on*(a) is the set of simple actions $\{a_1, \ldots, a_n\}$ in A_i, which directly or indirectly provide a with a service; i.e., for each action a_h in $\{a_1, \ldots, a_n\}$ there exists a sequence of causal links in C_i from a_h to a. Possibly, *depends_on*$(a) = \emptyset$ only when a is a target action.

Distributed Control Loop. As stated in the introduction, we propose a distribute approach where each agent i of the team $\mathcal{T}$ is responsible for monitoring and diagnosing the actions it performs summarized in the architecture reported in Figure 1. The plan instance P, achieving a desired complex goal is synthesized by a human user possibly by exploiting planning tools (such as (Boutilier & Brafman 2001)). Given the global P, the Dispatcher module decomposes P in as many sub-plans as the agents in $\mathcal{T}$. The decomposition is an easy task, which involves just the selection from P of all the actions an agent has to execute; formally, the sub-plan for agent i is the tuple P_i=$\langle A_i, <_i, C_i, X_i^{in}, X_i^{out}\rangle$ where: A_i is the subset of actions in A that agent i has to execute; $<_i$ is a total order relation defined over the actions in A_i; C_i is a set of causal links $a \stackrel{q}{\rightarrow} a'$ where both a and a' belong to A_i; X_i^{in} is a set of *incoming* causal links where $a' \in A_i$ and $a \in A_j$; finally, X_i^{out} is a set of *outgoing* causal links.

After this initial phase, each agent starts the execution of its own sub-plan. In particular, each agent performs a local control loop on the progress of its actions. This local control loop involves the Plan Execution Monitoring and Diagnosis module (PEMD), the Fault Recovery module (FR) and the Local Re-plan module (LP). First of all, the PEMD module is responsible for estimating the current status of the agent and for detecting the *outcome* of the actions the agent is executing. To this aim the PEMD exploits the observations obs_t^i coming from the environment.

Every time agent i detects a not nominal outcome, it tries to recover from this failure by activating a local re-planning phase aimed at synthesizing a local recovery plan. In particular, in the current architecture of the supervisor, the FR module coordinates the recovery process: first, the FR invokes the LP module, which has to infer a new local plan that agent i can execute to overcome the failure. In case such a local plan does not exist, agent i notifies the human user about the detected action failure.

MOVE(A, P1, P2)	LOAD(A, OBJ, P)	PUT_DOWN(A, OBJ, P)
pre: AT(A, P1)	**pre**: AT(A, P)	**pre**: AT(A, P)
FREE(P2)	AT(OBJ, P)	LOADED(A, OBJ)
	EMPTY(A)	
eff: ∼AT(A, P1)	**eff**: ∼EMPTY(A)	**eff**: ∼LOADED(A, OBJ)
∼FREE(P2)	∼AT(OBJ, P)	AT(OBJ, P)
AT(A, P2)	LOADED(A, OBJ)	EMPTY(A)

Table 1: The preconditions and the effects of the actions in the plan of Figure 2.

While in (Micalizio, Torasso, & Torta 2006a) we have addressed the FD and the LP modules, which are aimed at the plan recovery from an action failure; in this paper we focus our attention on the PEMD module, devoted to the monitoring and the diagnosis of the execution of the local plan P_i. Observe that, since actions may be causally dependent, the PEMD modules of different agents may need to communicate during the plan supervision process (see the communication channel between the PEMD modules in Figure 1).

Running Example. In the paper we will use a simple example from the blocks world for illustrating the concepts and the techniques introduced for monitoring and diagnosing the execution of a MAP. Let us consider two agents, `A1` and `A2`, which have to cooperate in order to achieve a specific configuration where the block `B1` is put on the block `B2`. To get this goal, agent `A1` has to move block `B1` from the source `S1` to the target `T1` by passing through door `D1`, while agent `A2` has to move block `B2` from `S2` to `T1` by passing through the doors `D2` and `D1`.

The use of the resources (source and target locations and doors) is constrained since a resource can be accessed by only one agent per time. However, there exists a further location, called `PRK` (i.e. parking area), which is not constrained and where the agents are positioned when they complete their sub-plans.

Table 1 reports the STRIPS-like definition of a sub-set of actions the agents can execute. For example, the action `LOAD(A,OBJ,P)` means that the agent `A` loads on board the object `OBJ` located at position `P`. The preconditions of this action require that both `A` and `OBJ` are located at `P` and that the agent is not loaded yet with another object. The effects of action `LOAD(A,OBJ,P)` state that the object is loaded on `A`.

Figure 2 shows the global plan produced by a POMP-like planner to achieve the target configuration of blocks, i.e. the global goal. The plan is a DAG where nodes correspond to actions and edges correspond to temporal (dashed) or causal (solid) links. The causal links are labeled with the services an action provides to another one, for example the causal link from action 1 to action 4 is labeled with the service `LOADED(A1,B1)`, which is both one of the effects of action 1 and one of the preconditions for the execution of action 4. It is easy to see that when action 1 fails, action 4 fails too since one of its preconditions is not satisfied.

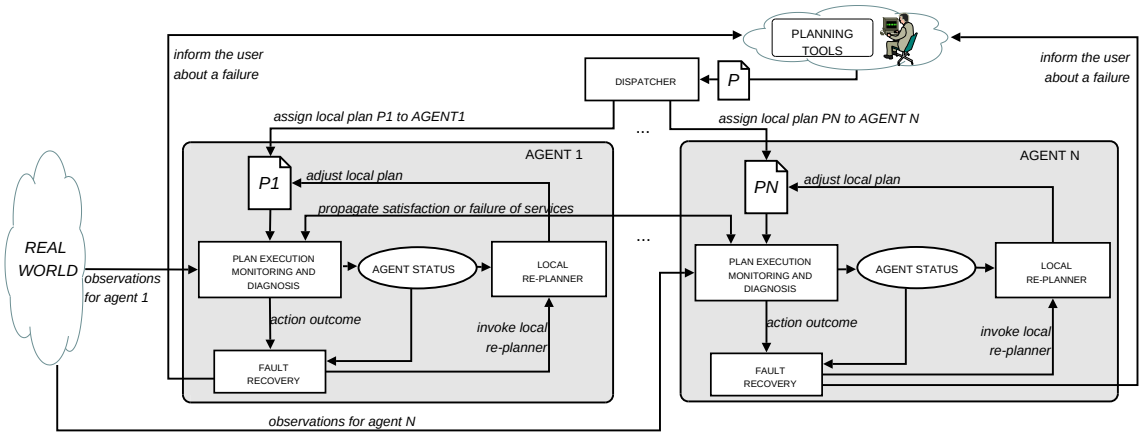

Figure 1: The architecture of the distributed control loop.

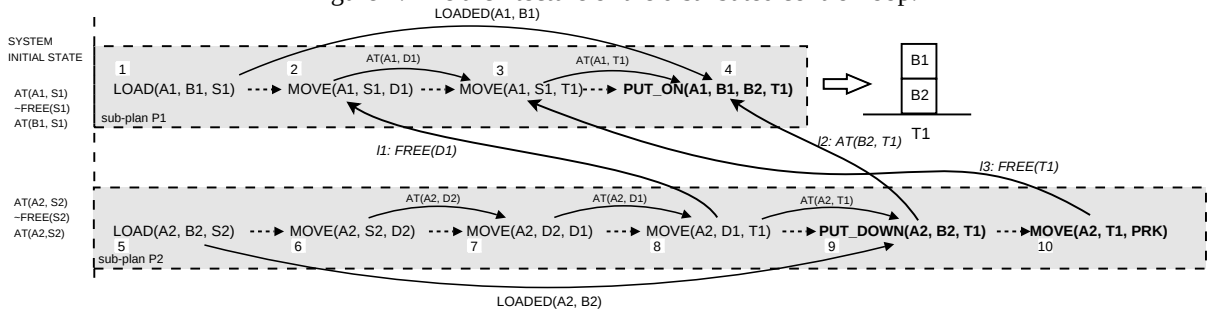

Figure 2: The plan built by a POMP-like planner.

In Figure 2 the target actions are in bold, for example given the target action 4, the set *depends_on*(4) is $\{1, 2, 3\}$. The grey, dashed rectangles highlight the sub-plans in which the global plan has been decomposed. The presence of the causal links l_1, l_2 and l_3 indicates that the sub-plans P_1 and P_2 have causal dependent actions.

Plan Execution Monitoring: Basic Concepts

Since each agent i has to supervise the actions it is responsible for, agent i has to determine, at each time t, the outcome of the action a_t (i.e., the action executed by i at time t), and its own health status after the execution of a_t.

Agent Status. The status of agent i is expressed in terms of a set of status variables VAR^i; in particular, this set of variables is partitioned into two subsets: $HEALTH^i$ and $ENDO^i$. $HEALTH^i$ denotes the set of variables concerning the health status of an agent functionalities (e.g., mobility and power), these variables are not directly observable, so their actual value can be just estimated. Whereas, $ENDO^i$ denotes the set of all the other endogenous status variables (e.g., the agent's *position*). The values assumed by the variables in $ENDO^i$ can be directly observed by agent i; however, since agent i receives just partial observations about the changes occurring in the system, at each time instant t the agent i observes the value of just a subset of variables in $ENDO^i$. Therefore, agent i is in general unable to precisely determine its own status; rather i can determine just a set of states after the execution of a_t, this set is known in literature as *belief state* and will be denoted as $\mathcal{B}_t^i$.

Extended Action Model. As mentioned above, the synthesis of the plan P can be performed by exploiting STRIPS-like action models, which consider only the nominal behavior of the actions. However, in order to supervise the execution of the actions in P_i, the agent i must have at disposal extended action templates, which model not only the nominal behavior of the actions, but also the actions behavior when faults occur. For such a modeling task we adopt a relational representation, which has been proved to be useful for the on-line monitoring and diagnosis of multi-agent systems (see (Micalizio, Torasso, & Torta 2006b)). In particular, this formalism is able to capture non deterministic effects of the action execution.

The extended model of an action a_t is a transition relation $\Delta(a_t)$, where every tuple $d \in \Delta(a_t)$ models a possible change in the status of agent i, which may occur while i is executing a_t. More precisely, each tuple d has the form $d = \langle s_{t-1}, fault, s_t \rangle$; where s_{t-1} and s_t represent two agent states at time $t-1$ and t respectively, each state is a complete assignment of values to the status variables VAR^i of agent i; $fault$ denotes the fault which occurs to cause a change of status from s_{t-1} to s_t (of course, in the transitions which model the nominal behavior $fault$ is empty). As commonly assumed in approaches to the diagnosis of Discrete Events Systems (see e.g. (Pencolé & Cordier 2005)), we assume that two faults can not occur simultaneously on the functionalities of the same agent. It follows that an agent

i can be struck by only one fault per time instant, however it can be affected by more than one faults over the time; moreover, the assumption allows the simultaneous occurrence of faults in different agents.

Table 2 shows the extended model of a MOVE action from a location $P1$ to a location $P2$; due to space reasons we show just a subsets of tuples of the transition relation $\Delta(\texttt{MOVE})$. Moreover, for sake of readability, the agent states are expressed just in the subset of status variables directly affected by the MOVE action, in particular they are: *position*, the position of the agent; *carried* the carried load (if any) by the agent, this variable can be either *empty* or *loaded*; *pwr*, the power of the agent's battery, which can be *high* (the nominal mode) or *low* (a degraded mode) and *mobility*, the health status of the agent's mobility functionality which can be *ok* (nominal) or *bk* (i.e., broken, a faulty mode). Observe that the STRIPS model of the MOVE action just represents the nominal behavior of MOVE, assuming that both power and mobility are in their nominal mode; the corresponding extended model represents also how the MOVE action behaves in any combination of power and mobility modes, included the faulty ones.

The nominal behavior of the MOVE action is modeled by transition 1, the $*$ symbol (don't care) indicates that, when the agent's functionalities behave nominally, the actual carried load does not impact the outcome of the action. In our example the MOVE action can fail as a consequence of two types of faults: f-BRY and f-MOB. f-BRY affects the battery by reducing the level of power from the nominal *high* to the degraded *low*. An agent can complete a move action with battery power *low* iff the agent is not carrying any block, the action fails otherwise (see transitions 2 and 3). f-MOB affects the health status of the mobility functionality, which changes from the nominal *ok* to the anomalous *bk* (broken); under this health status there is no way to complete the move action (see transition 4).

Action Outcome. Since the actual execution of an action is threatened by the occurrence of unexpected events (e.g., faults), the outcome of an action may be not nominal. The outcome of an action is *succeeded* when all the effects of the action have been achieved, or *failed* otherwise. Since at each time instant t we may have to deal with an ambiguous belief states $\mathcal{B}_t^i$, we consider the action a as succeeded only when all the expected effects of action a hold in every state s included in $\mathcal{B}_t^i$; more formally.

Definition 1 *The outcome of action a_t, executed by i at time t, is* succeeded *iff $\forall q \in eff(a_t)$, $\forall s \in \mathcal{B}_t^i$, $s \models q$; i.e., iff all the atoms q in $eff(a_t)$ are satisfied in every state s in $\mathcal{B}_t^i$.*

Of course, when we can not assert that action a_t is succeeded we assume that the action is failed.

Agent Trajectory. One of the results of the monitoring performed by agent i over the execution of its actions is a trajectory of the status of i itself over time. In general, the trajectory of a system is the sequence of states, actions and observations generated from a given initial state to some time point of interest. However, since the system is only partially observable, agent i can not maintain the actual trajectory of its own state; rather agent i maintains a set of possible trajectories. In the following $Tr_i[0,t]$ will denote the set of trajectories tr maintained by i in the interval $[0,t]$.

Monitoring with uncertain action outcomes

In this section we describe the methodology for monitoring the execution of a MAP when the outcome of some actions may be uncertain. The algorithm implementing the approach is reported in Figure 3. Due to space reasons, we focus just on the basic concepts the algorithm relies on.

The prediction process. In order to monitor the execution of action a_t, the first step performed by agent i is to extend the set of trajectories $Tr_i[0,t-1]$ with all possible state transitions of action a_t. In particular the prediction process is formalized in terms of the relational operators as:

Definition 2 *Given the set of trajectories $Tr_i[0,t-1]$ and the extended model of the action a_t, Tr_i is incrementally extended as:* $Tr_i[0,t]=\sigma_{obs_t^i}(Tr_i[0,t-1] \bowtie \Delta(a_t))$

The join operator $Tr_i[0,t\text{-}1] \bowtie \Delta(a_t)$ represents the prediction step, it has the effect of adding, to each trajectory $tr \in Tr_i[0,t-1]$, a possible state of agent i after the execution of action a_t. This set of predictions is filtered out w.r.t. the observations available at time t by means of the selection $\sigma_{obs_t^i}$ (see line 8 of the algorithm of Figure 3).

Observe that $Tr_i[0,t]$ can be extensionally written as:
$\sigma_{obs_t^i}(\sigma_{obs_{t-1}^i}(\ldots(\sigma_{obs_1^i}(\mathcal{B}_0^i \bowtie \Delta(a_1))) \ \ldots \ \bowtie \ \Delta(a_{t-1})) \bowtie \Delta(a_t))$
where $\mathcal{B}_0^i$ represents the initial belief state of agent i.

Given $Tr_i[0,t]$, agent i is able to extract by means of the relational projection the current belief state $\mathcal{B}_t^i$; in this way agent i evaluates the outcome of action a_t according to Definition 1; formally, $\mathcal{B}_t^i=\pi_t(Tr_i[0,t])$.

Monitoring and system observability. If the system were completely observable, the predicted belief state $\mathcal{B}_t^i$ would contain just the actual status of agent i at time t. Of course, this condition is rather unrealistic and, in general, one has to deal with uncertain belief states. On the contrary, some minimal requirements on the system observability have to be met to provide agent i with the possibility of evaluating the outcome of the actions it has to execute.

Basically we consider two possible levels of system observability: *strong* and *weak*. Strong observability corresponds to the assumption that agent i is able to determine the outcome of every action a_t it executes; i.e., the belief state $\mathcal{B}_t^i$, filtered out w.r.t. obs_t^i, is sufficiently precise for testing the condition of Definition 1.

Dealing with uncertain outcomes. Unfortunately, the conditions of strong observability do not hold in many domains. In the general case, the estimated belief state $\mathcal{B}_t^i$ is not sufficiently precise to determine the outcome of action a_t. In other words, it may be possible that the effects of a_t are satisfied in just a not empty subset of states in $\mathcal{B}_t^i$.

In this case agent i has to keep track of the actions it has executed and whose outcome has not been determined yet. In the following, $pendingOutcomes_i(t)$ denotes the set of actions, so far executed by agent i, whose outcome has not been determined at time t. In other words, at each time instant t, either agent i is able to determine the outcome of a_t or a_t is included in the set $pendingOutcomes_i(t)$.

	s_{t-1}				$fault$	s_t			
	endogenous		**health variables**			**endogenous**		**health variables**	
	position	*carried*	*power*	*mobility*		*position*	*carried*	*power*	*mobility*
1	P1	*	High	OK	*null*	P2	*	High	OK
2	P1	EMPTY	High	OK	f-BRY	P2	EMPTY	Low	OK
3	P1	LOADED	High	OK	f-BRY	P1	LOADED	Low	OK
4	P1	*	High	OK	f-MOB	P1	*	High	BK

Table 2: The extended model of action MOVE(A,P1,P2): the transition relation $\Delta(MOVE)$.

Plan-Monitoring-and-Diagnosis($\mathcal{B}_0^i$, P_i) [P_i=$\langle A_i, <_i, C_i, X_i^{in}, X_i^{out}\rangle$] {
01 t = 0; t_s = 0; $Tr_i[t_s, t] = \mathcal{B}_0^i$; $pendingOutcomes_i(t) = \emptyset$; *nextAction* = null;
02 **while** (true){ $t = t + 1$;
03 **if** (*nextAction*=null) *nextAction* = ⟨*get next action from* P_i⟩;
04 **if** (*all incoming links of nextAction are marked as satisfied*) mark *nextAction* as *executable*;
05 **else** ⟨ execute *WaitAction* ⟩;
06 **if** (*nextAction* is *executable*) {
07 a_t=*nextAction*; *nextAction*=null; ⟨execute a_t⟩; obs_t^i = ⟨*get current observations for agent i*⟩;
08 $Tr_i[t_s, t]$=$\sigma_{obs_t^i}(Tr_i[t_s, t\text{-}1] \bowtie \Delta(a_t))$=$\sigma_{obs_t^i}(\sigma_{obs_{t-1}^i}(\ldots(\sigma_{obs_{t_s+1}^i}(\mathcal{B}_{t_s}^i \bowtie \Delta(a_{t_s+1}))) \ldots \bowtie \Delta(a_{t-1})) \bowtie \Delta(a_t))$;
09 $pendingOutcomes_i(t) = pendingOutcomes_i(t-1) \cup \{a_t\}$;
10 $\langle completedActs_i, primaryFailedActs_i \rangle$=**EvaluateOutcomes**($pendingOutcomes_i(t)$, $Tr_i[t_s, t]$);
11 **for** (each action $a \in completedActs_i$){
12 **for** (each link $l : a \xrightarrow{q} a' | l \in C_i$) mark l as *satisfied*;
13 **for** (each link $l : a \xrightarrow{q} a' | l \in X_i^{out}$) notify *agent*($a'$) the satisfaction of link l; }
14 $AgentDiagnosis_i = \emptyset$; $secondaryFailedActs_i$=$\emptyset$; *Inbox*=⟨get incoming messages⟩;
15 **for** (each satisfied link $l : a' \xrightarrow{q} a''$, $l \in$ *Inbox* | $l \in X_i^{in}$) mark l as *satisfied*;
16 **if** ($primaryFailedActs_i \neq \emptyset$) {
17 *ACTS* = $\emptyset$; *LNKS* = $\emptyset$; $AgentDiagnosis_i$ = **getAgentDiagnosis**($primaryFailedActs_i$, $Tr_i[t_s, t]$);
18 **for** each action $a \in primaryFailedActs_i$ {
19 **Propagate**(a, P_i, *ACTS*, *LNKS*); $secondaryFailedActs_i = secondaryFailedActs_i \cup ACTS$
20 **for** (each failed link $l' : a'' \xrightarrow{q} a'''$, $l' \in$ *LNKS*) notify *agent*(a''') the failure of l'; } } }
21 **repeat**
22 changes = false; *Inbox*=⟨get incoming messages⟩;
23 **for** (each failed link $l : a' \xrightarrow{q} a''$, $l \in$ *Inbox* | $l \in X_i^{in}$){ *LNKS*=$\emptyset$; *ACTS*=$\emptyset$; changes = true;
24 **Propagate**(a'', P_i, *ACTS*, *LNKS*); $secondaryFailedActs_i = secondaryFailedActs_i \cup ACTS$
25 **for** (each failed link $l' : a'' \xrightarrow{q} a'''$, $l' \in$ *LNKS*) notify *agent*(a''') the failure of l'; }
26 **until** ($\sim$ changes)
27 **if** ($primaryFailedActs_i \neq \emptyset$) $\vee$ ($secondaryFailedActs_i \neq \emptyset$)
28 **return** ⟨ $AgentDiagnosis_i$, $primaryFailedActs_i$, $secondaryFailedActs_i$⟩;
29 **else if** (a_t *is a target action*) { $pendingOutcomes_i(t) = \emptyset$; $\mathcal{B}_t^i = \pi_t(Tr_i[t_s, t])$; $t_s = t$; $Tr_i[t_s, t] = \mathcal{B}_t^i$ } } }

Figure 3: The algorithm for monitoring and diagnosing the execution of a local plan.

It is worth noting that the outcome of a_t can be exploited to determine the outcome of some actions in $pendingOutcomes_i(t)$, in fact the following property holds.

Property 1 *Let a_t be an action (either target or simple) with outcome* succeeded*, then all the actions a_h in $pendingOutcomes_i(t) \cap depends\_on(a_t)$ have outcome* succeeded *too.*

However, when an action a_t fails, agent i may be unable to determine whether failure is due to a fault during the execution of a_t or a_t is failed as a consequence of the failure of other actions i in *depends_on*(a_t). In this case, in order to be conservative, we introduce the following policy.

Policy 1 *Let a_t be an action (either target or simple) with outcome* failed*, then all the actions a_h in $pendingOutcomes_i(t) \cap depends\_on(a_t)$ are marked* failed *too.*

At line 10 of the algorithm, function **EvaluateOutcomes** evaluates the outcomes of the actions in $pendingOutcomes_i(t)$ and returns the pair $\langle completedActs_i, primaryFailedActs_i \rangle$, where in $completedActs_i$ ($primaryFailedActs_i$) the *succeeded* (*failed*) actions are collected.
In any case, whenever agent i determines the outcome of an action a, it has to propagate the effects of this outcome in the global plan according to the following policies:

Policy 2 Communicate Positive Information *if a is* succeeded*, $\forall l : a \xrightarrow{q} a' | l \in X_i^{out}$, notify $agent(a')$ that the service q has been provided.*

Policy 3 Communicate Negative Information *if a is* failed*, $\forall l : a \xrightarrow{q} a' | l \in X_i^{out}$, notify $agent(a')$ that the service q can not be provided.*

Policies 2 and 3 are implemented, respectively, in lines 11-13 and 18-20 of the algorithm.
Running Example. Let us assume that in the blocks world example, under weak observability, the LOAD actions are

not observable; moreover the MOVE actions are observable only when an agent gets the target T1 or the parking area PRK. Thereby, the outcome of actions 1, 2, 5, 6, 7 can not be determined just relying on the system observations. Let us assume that at time t, agent 2 executes action 8. Observe that at time t, $pendingOutcomes_2(t)$=$\{5, 6, 7\}$; while $pendingOutcomes_1(t)$=$\{1\}$. If the outcome of action 8 is *succeeded*, agent 2 determines that actions 6 and 7 are *succeeded* too; moreover it informs agent 1 that the service *free(D1)* has been provided. Notice that, to determine the outcome of action 5, agent 2 needs to know the outcome of the target action 9. On the contrary, if the outcome of action 8 is failed, agent 2 informs agent 1 that service *free(D1)* can not be provided and activates the diagnostic process.

Agent Diagnosis and Plan Diagnosis

The presence of causal links between actions introduces dependencies even among different sub-plans. As a consequence, the failure of an action (*primary failure*) may affect many other actions in the global plan (*secondary failures*).

The distinction between primary and secondary failures is an important piece of information, which must be considered in the notion of *plan diagnosis*. In fact, in order to recover from the failure of a set of actions in the plan, one needs to know: 1) which actions are the primary failures and 2) which combination of faults in the functionalities of the agents may have caused these failures, i.e. the *root causes* of these failures.

Agent Diagnosis. As described in the previous section, as soon as agent i detects the failure of action a_t, i starts a diagnostic reasoning step in order to determine the root causes of this failure (see line 17 of the algorithm). It is worth noting that, the failure of action a_t may be a consequence of 1) the occurrence of a fault during the execution of a_t; or 2) the failure of an action a_h, previously executed ($h < t$). In the latter case, the action a_h is included in the set $pendingOutcomes_i(t)$ and its (not nominal) outcome can not be unequivocally determined as the system is only partially observable.

When agent i is unable to distinguish between the two possible cases, i adopts a conservative behavior (according to Policy 1) and determines the primary failures as follows:

Definition 3 *Let a_t be an action executed by agent i at time t, let* failed *be the outcome of a_t. The set of primary failures related with the failure of a_t is the set of actions: $primaryFailedActs_i$= $(pendingOutcomes_i(t) \cap$ depends_on$(a_t)) \cup a_t$.*

Observe that action a_t is always included in the set of the primary failures.

Once the primary failures have been singled out, agent i has to infer a local diagnosis, where the root causes of each primary failure are highlighted.

Definition 4 *Let $a_h \in primaryFailedActs_i$ be the action executed at time h. The root causes of the failure of a_h is the set of faults $faults(a_h)$=$\pi_{faults}(\mathcal{B}_h^i \bowtie \Delta(a_h))$*

In this case the join $\mathcal{B}_h^i \bowtie \Delta(a_h)$ is an abductive step, since the relation $\Delta(a_h)$ is not used to predict a new belief state (as in the prediction process in Definition 2), but it is used for inferring the previous belief state $\mathcal{B}_{h-1}^i$. The projection π_{fault} collects the possible faults which may have caused a transition from a state in $\mathcal{B}_{h-1}^i$ to a state in $\mathcal{B}_h^i$, i.e., the result of the collection is the set of faults which may have occurred during the execution of action a_h. Since the failure of a_t can be explained by the failure of just one action in $primaryFailedActs_i$, the agent diagnosis for i is defined as:

Definition 5 *Given the set $primaryFailedActs_i$, the diagnosis for agent i $AgentDiagnosis_i$ is defined as $AgentDiagnosis_i = \bigvee_{a_h \in primaryFailedActs_i} faults(a_h)$*

In other words, $AgentDiagnosis_i$ is a disjunction of faults $f_1 \vee f_2 \vee \ldots \vee f_n$ affecting some functionalities of i, where each fault f_h is sufficient to explain the failure of an action in $primaryFailedActs_i$ and, as a consequence, of the action a_t. The inference of the agent diagnosis is part of the **getAgentDiagnosis** function (line 17).

Plan Diagnosis. Now we need to determine at what extent the primary failures affect the global plan P. In particular, we aim at singling out which actions in P won't become *executable* as a direct or indirect consequence of the failure of an action in $primaryFailedActs_i$. To this end we exploit the set C of causal links.

Definition 6 *Given the global plan P=$\langle A, <, C\rangle$ and the set $primaryFailedActs_i$ determined by agent i, the set of secondary failures is*
$secondaryFailedActs_i$=$\{a \in A$ *such that*

$\exists$ *a causal link* $l' : a' \xrightarrow{q'} a$, *where* $l' \in C$ *and* $a' \in primaryFailedActs_i$ *or*

$\exists$ *a causal link* l" $: a$" $\xrightarrow{q"} a$, *where* l" $\in C$ *and* a" $\in secondaryFailedActs_i\}$

Observe that, in order to single out the secondary failures in whole plan, the agents in the team have to cooperate. In particular, the propagation of a failure in the global plan is an iterative process in which the agents communicate one another the set of services which can longer not be provided. In the algorithm the process for computing the secondary failures is encoded in lines 21-26 of the algorithm and it is based on the **Propagate** function described in (Micalizio, Torasso, & Torta 2006a). In definitions 5 and 6 we have introduced the notion of plan diagnosis w.r.t. an agent i; however, these definitions can be extended to the more general case where the plan diagnosis considers all the agents in the team $\mathcal{T}$. More precisely, given the global plan P, it is possible to demonstrate that the plan diagnosis for P can be computed in distributed way as the pair $\langle primaryFailedActs, secondaryFailedActs\rangle$ where $primaryFailedActs = \bigcup_{i \in \mathcal{T}} primaryFailedActs_i$ and $secondaryFailedActs = \bigcup_{i \in \mathcal{T}} secondaryFailedActs_i$.

Correctness. The correctness analysis of the algorithm has to consider the prediction process and the inferences for computing the agent and plan diagnosis. Regarding the prediction process it is possible to demonstrate that the belief state $\mathcal{B}_t^i$, estimated at each time t, always contains the actual status of the agent i, i.e., the monitoring keeps track of the actual status of the agents. As concerns the agent diagnosis $AgentDiagnosis_i$, it always includes the actual set of faults which have caused a detected action failure

(this follows directly from the properties of the belief state $\mathcal{B}_t^i$). Finally, considering the plan diagnosis, it is easy to see that: 1) the set $primaryFailedActs_i$ actually contains all and only the actions whose failure may have caused the detected anomaly in the execution of the sub-plan P_i; and 2) the set $secondaryFailedActs_i$ includes all and only the actions in the global plan P which are affected by the failure of some actions in $primaryFailedActs_i$.

Moreover, assuming that the system observability is sufficient to determine unequivocally the outcome of every target action in the global plan P; and assuming that in each sub-plan P_i, the distance (in number of actions) between two consecutive target action is less than or equal to a constant k; it is possible to demonstrate that each agent $i \in \mathcal{T}$ is able to determine the outcome of all the actions it executes within a finite time window. This property allows agent i to maintain trajectories including just the last k time instants (in the algorithm, agent i maintains the trajectories in the interval $[t_s, t]$, where t_s is the time when the last target action has been successfully executed).

Running Example. Under the same conditions previously described, let us assume that at time t the failure of action 8 activates the diagnostic step. First of all, agent 2 determines the set of primary failures $primaryFailedActs_2 = \{6, 7, 8\}$. Then, agent 2 infers its local diagnosis, in this simple example $AgentDiagnosis_2$=$\{$`f-BRY` $\vee$ `f-MOB`$\}$. Finally, agent 2 propagates the harmful effects of the primary failures in the global plan in order to detect the secondary failures. In particular, the set $secondaryFailedActs_2$=$\{2, 3, 4, 9\}$ is obtained by means of the cooperation of agent 1, which propagates the failure of the causal link $l1$ in its own sub-plan.

Conclusions

While the problems of distributed monitoring and diagnosis of component-based systems have been deeply discussed in literature (see e.g., (Pencolé & Cordier 2005)); just a few solutions have been proposed in the multi-agent setting. In (Kalech & Kaminka 2005) the concept of *social diagnosis* is introduced; in particular they consider multi-agent systems where each agent chooses, at each time, the more appropriate behavior to assume. The social diagnosis has to explain the disagreements among cooperating agents (i.e., when some agents assume incompatible behaviors).

The approach presented in the paper is similar to the one described in (Witteveen *et al.* 2005), where Witteveen et al. propose a distributed approach to monitoring and diagnosing the execution of a MAP. This approach adopts an *object* (or *resource*) view since the status of the system is represented in terms of the status of the system objects. The approach assumes that every agent monitors and diagnoses every action it is responsible for. Actions are atomic and are modeled as functions of their nominal behavior only, whereas the anomalous behavior of the actions is not explicitly modeled. As a consequence, whenever an action failure is detected, the status of a subset of resources can not be predicted and the monitoring is incomplete.

In this paper, we have proposed a distributed approach for monitoring and diagnosing the execution of a multi-agent plan in a system which is only partially observable. In particular, while in (Micalizio, Torasso, & Torta 2006a) we have discussed a monitoring approach which assumes to determine the outcome of every action exploiting the available observations (strong observability) in this paper we have discussed an extension which is able to deal with uncertain action outcomes (weak observability).

Differently from the approach by (Witteveen *et al.* 2005), we introduce an extended model of actions for capturing both nominal and anomalous execution. Thereby the monitoring process we propose is complete since the status of the system is estimated even after the occurrence of a fault.

Moreover, while in (Witteveen *et al.* 2005) the notion of diagnosis refers just to the pair of primary and secondary failures, the use of extended action models allows us to introduce the notion of diagnosis which includes the root causes of the detected primary failures. Observe that the root causes are essential in order to recover from a failure, in fact, as discussed in (Micalizio, Torasso, & Torta 2006a), a local re-planner needs to know the current health status of an agent in order to establish which actions can or can not be used in the recovery plan. The extended model of the actions are encoded via OBDDs (Ordered Binary Decision Diagrams) and all the relational operations involved in the algorithm are performed by means of standard operations on OBDDs, according to the approach described in (Micalizio, Torasso, & Torta 2006b).

References

Birnbaum, L.; Collins, G.; Freed, M.; and Krulwich, B. 1990. Model-based diagnosis of planning failures. In *Proc. AAAI'90*, 318–323.

Boutilier, C., and Brafman, R. I. 2001. Partial-order planning with concurrent interacting actions. *JAIR* 14:105–136.

Carver, N., and Lesser, V. 2003. Domain monotonicity and the performance of local solutions strategies for cdps-based distributed sensor interpretation and distributed diagnosis. *Jour. of Aut. Agents and MAS* 6:35–76.

Kalech, M., and Kaminka, G. 2005. Diagnosing a team of agents: Scaling up. In *Proc. AAMAS'05*, 249–255.

Kurien, J., and Nayak, P. 2000. Back to the future for consistency-based trajectory tracking. In *Proc. American Association for Artificial Intelligence (AAAI00)*, 370–377.

Micalizio, R.; Torasso, P.; and Torta, G. 2006a. Intelligent supervision of plan execution in multi-agent systems. *Int. Trans. on Systems Science and Applications* 1(3):259–267.

Micalizio, R.; Torasso, P.; and Torta, G. 2006b. On-line monitoring and diagnosis of a team of service robots: a model-based approach. *AI Comm.* 19(4):313–349.

Pencolé, Y., and Cordier, M. 2005. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence* 164:121–170.

Witteveen, C.; Roos, N.; van der Krogt, R.; and de Weerdt, M. 2005. Diagnosis of single and multi-agent plans. In *Proc. AAMAS05*, 805–812.