Agent Cooperation for Monitoring and Diagnosing a MAP

Roberto Micalizio and Pietro Torasso

Dipartimento di Informatica, Università di Torino, Torino, Italy
{micalizio,torasso}@di.unito.it

Abstract. The paper addresses the tasks of monitoring and diagnosing the execution of a Multi-Agent Plan, taking into account a very challenging scenario where the degree of system observability may be so low that an agent may not have enough information for univocally determining the outcome of the actions it executes (i.e., pending outcomes).

The paper discusses how the ambiguous results of the monitoring step (i.e., trajectory-set) are refined by exploiting the exchange of local interpretations between agents, whose actions are bounded by causal dependencies. The refinement of the trajectory-set becomes an essential step to disambiguate pending outcomes and to explain action failures.

1 Introduction

The problem of supervising the execution of a multi-agent plan (MAP) is receiving an increasing attention; in fact, the idea of distributing the execution of a complex plan among a number of cooperating agents, which execute actions concurrently, has proved to be quite useful for several domains and applications. The supervision is a complex task as one has to take into account *plan threats*, which can cause action failures. In the last few years some approaches have been proposed to attack the problem ([1,2,3]). Typically these approaches assume that action failures are not consequences of plan flaws, but failures are due to the occurrence of exogenous events (such as unexpected changes in the environment, occurrence of faults in some agents functionalities, etc.). Moreover, because of causal dependencies between actions executed by different agents, the failure in an action assigned to an agent may impact also the execution of the actions assigned to other agents. For this reason, it is necessary to perform a plan diagnosis in order to detect an action failure as soon as possible, and to single out (if possible) the reason for such a failure. In fact, the ability of an agent to perform some form of the plan recovery and repair strongly depend on the capabilities of inferring a precise diagnosis (see for example [4]).

In this paper we advocate a distributed approach to plan supervision, where each agent is responsible for supervising (monitoring, detecting action failures and performing plan diagnosis) the actions it executes. In particular, action models represent not only the nominal action behavior, but also the (usually non deterministic) anomalous behavior due to the occurrence of exogenous events. Of course, the adoption of non deterministic action models make the supervision task even more complex.

Moreover, since the system is distributed, each agent has just a limited view of the progress of the global plan; in fact it receives only partial observations from the environment. As a consequence an agent cannot, in general, precisely detect the outcome (success or failure) of its actions on the sole basis of the observations it receives. The proposed solution involves the exchange of local interpretations of plan execution (i.e., action outcomes) among agents, in order to refine the local point of view of each agent, and possibly to detect and explain action failures.

The paper is organized as follows: first, 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 communication and cooperation among agents for inferring the action outcomes which cannot be immediately determined.

2 Distributed Execution of a Multi-Agent Plan

In this paper we consider a specific class of multi-agent systems which can be modelled as Multi-Agent Plan (MAP). In a MAP, a team $\mathcal{T}$ of agents strictly cooperate to reach a common, complex goal G by exchanging one another services and this cooperative behavior introduces causal (and precedence) dependencies among their activities.

Global plan. The MAP P is modeled as the tuple $\langle A, E, CL, CC, NC \rangle$ (see e.g., [5]) such that: A is the set of the action instances a the agents have to execute, each action a is assigned to a specific agent in the team; E is a set of precedence links between action instances, 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: a pair $\langle a, a' \rangle$ in CC models a joint action, while constraints in NC prevent conflicts for accessing the resources.

To keep the discussion simple, in this paper we do not consider joint actions even though the approach can be extended to deal with them (see e.g. [3]). Moreover, we translate the non-concurrency constraints into precedence links, so that during the plan execution agents do not need to negotiate for accessing resources (this is equivalent to the *concurrency requirement* introduced in [2]); in particular, each non concurrency constraint $\langle a, a' \rangle \in NC$ (ruling the mutual exclusion access to a resource res), is substituted either with $a \prec_{res} a'$ or with $a' \prec_{res} a$. The result of this translation procedure is a simple form of scheduling, where actions assigned to different agents are explicitly related by means of a (partial) precedence relation; formally, the MAP P to be executed (and supervised) is defined as $P{=}\langle A, E, CL, RE \rangle$, where RE is the set of precedence links ruling the access to the resources.

Local Plans. Since the MAP P is executed in a distributed way by the agents in the team, each single agent is responsible just for a portion of a MAP called *local plan*. Intuitively, a local plan P^i is the projection of the global plan P over the action instances assigned to the agent i. Thus, the plan P is decomposed into as many local plans as the agents in $\mathcal{T}$; however, the decomposition must keep trace of the causal and precedence relations existing between action instances assigned to different agents: the local plan P^i for agent i is formally defined as the tuple $P^i{=}\langle A^i, E^i, CL^i, T^i_{in}, T^i_{out}, RE^i_{in}, RE^i_{out}\rangle$, where A^i, E^i and CL^i have the same meaning of the sets A, E and CL, respectively, restricted to actions assigned to agent i. The remaining sets are used to keep a trace of the dependencies existing between actions belonging to different local plans; in particular, the causal links from (to) an action of agent i to (from) an action of another agent j are collected in the sets T^i_{out} (outgoing links) and T^i_{in} (incoming links), respectively. Similarly, the precedence links for accessing the resources are subdivided into RE^i_{in} (incoming links) and RE^i_{out} (outgoing links).

In order to simplify the discussion, we assume that each local plan P^i is *totally ordered*, thereby it can be represented as the sequence of actions $\langle a^i_0, a^i_1, \ldots, a^i_\infty\rangle$.

Distributed execution and coordination under nominal conditions. As an effect of the decomposition of the global plan, the execution of the local plans is performed by the agents concurrently and asynchronously. In particular, we assume an agent executes its next action a as soon as the preconditions of a are satisfied; on the contrary, the agent will wait as long as the preconditions of a are not satisfied.

Because of the causal and precedence constrains introduced by the planning step, the agents have to coordinate one another by exchanging messages. All the knowledge required for inter-agent communication is encoded in P^i: every outgoing causal (or precedence) link in T^i_{out} (RE^i_{out}) is associated with a *send-message* operation toward agent j. Similarly, every incoming causal (or precedence) link in T^i_{in} (or RE^i_{in}), is associated with a *receive-message* operation[1].

Let us suppose that a causal link in T^j_{in} states that the preconditions for action a (to be executed by agent j) involve a service q provided by agent i: the agent j will wait a message from i about the service q before executing a.

Plan threats. The nominal execution of the local plan may be affected by *plan threats*, which typically cause action failures. In this paper, plan threats are exogenous events which cause abrupt changes in the agent status; we will denote as $\mathcal{E}$ the set of exogenous events which may occur during the plan execution; while we will use the symbol ϵ to represent the absence of exogenous event.

Observe that communication and cooperation among agents are not only needed under nominal execution of the plan, but are even more important in presence of some failure. In fact, when agent i realizes that agent j cannot be provided with service q due to an action failure, it has to exploit the causal link in T^i_{out} to send agent j a message about the non availability of service q: agent j becomes aware that the preconditions of action a will never be satisfied.

The cooperation and communication becomes more critical if we want to deal with cases when the system observability is so partial that an agent i is unable to conclude that the service q has been provided or not, and therefore it cannot guarantee the agent j that q has been achieved.

[1] We assume that the inter-agent communication is reliable and instantaneous.

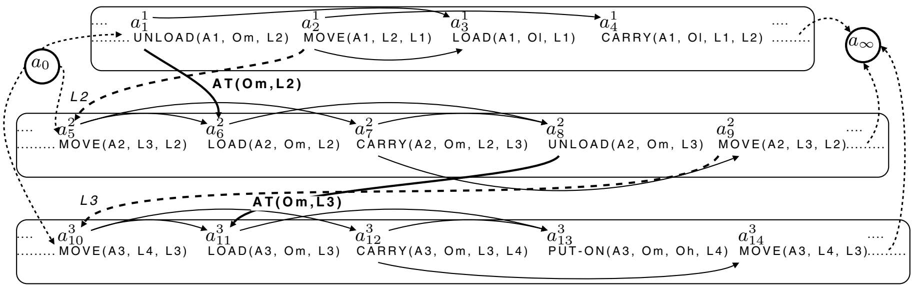
a_0
a_∞
a_1^1 UNLOAD(A1, Om, L2)
a_2^1 MOVE(A1, L2, L1)
a_3^1 LOAD(A1, Ol, L1)
a_4^1 CARRY(A1, Ol, L1, L2)
AT(Om,L2)
L2
a_5^2 MOVE(A2, L3, L2)
a_6^2 LOAD(A2, Om, L2)
a_7^2 CARRY(A2, Om, L2, L3)
a_8^2 UNLOAD(A2, Om, L3)
a_9^2 MOVE(A2, L3, L2)
L3
AT(Om,L3)
a_{10}^3 MOVE(A3, L4, L3)
a_{11}^3 LOAD(A3, Om, L3)
a_{12}^3 CARRY(A3, Om, L3, L4)
a_{13}^3 PUT-ON(A3, Om, Oh, L4)
a_{14}^3 MOVE(A3, L4, L3)

Fig. 1. The global plan to be monitored

In this way, agent j cannot be sure whether the preconditions of its next action a (which include q) are satisfied or not. If j adopted a conservative policy, it would not execute action a, and hence it would stop the execution of its local plan; however, this kind of policy may result impractical in many domains. We propose a weak commitment policy where we assume by default that service q has been provided, so the agent j can execute the action a. Obviously such a default assumption can be wrong in some cases and therefore the mechanisms for detecting the action failure and for performing plan diagnosis are much more complex when respect to the case no default assumption is made.

Running Example. Throughout the paper we will illustrate the proposed methodology by means a simple example from the blocks world. Let us consider three agents that cooperate to achieve a global goal G where a number of objects `O1`,..., `On` must be moved from a source location `L1` to a target position `L4`, passing through the intermediate positions `L2` and `L3`. All these positions are critical resources as only one agent can access one of them at a given time instant; moreover, these positions are the only locations where, under nominal conditions, an agent can pick up or release a block. Figure 1 shows a portion of a MAP achieving the goal G; in particular the picture shows the actions involved in the delivery of the object `Om`: each agent is responsible for carrying the object from a position to the next one: `A1` from `L1` to `L2`, `A2` from `L2` to `L3`, and `A3` from `L3` to `L4` where `Om` is put on the top of a stack of objects.

The three rectangles enclose the local plans assigned to the agents. It is easy to see that the MAP is a DAG where nodes are action instances and edges represent precedence or causal links. However, for the sake of readability, internal precedence links (i.e., between actions in the same local plan) are not displayed, while internal causal links (thin solid edges) are reported without labels. Instead, the picture highlights causal (solid edges) and precedence (dashed edges) links between actions in different local plans: these links represent relations between agents. For example, the causal link between actions a_1 and a_6 means that the agent `A1` provides agent `A2` with the service *at(Om, L2)* (i.e., the object `Om` is located in position `L2`); whereas the precedence link between actions a_2 and a_5, labeled with the resource id `L2` means that action a_5 can be executed only after the execution of action a_2, i.e., only when the resource L2 is no longer used by agent A1 and it is made available to agent A2.

3 Basic Concepts on Distributed MAP Monitoring

In this section we introduce the model-based methodology we adopt for monitoring the execution of a MAP. In particular, the models we propose for the agent state and actions take care of the inherent ambiguity of the system.

Agent state. Intuitively, the status of the system can be represented in terms of the status variables of the agents in the team $\mathcal{T}$ and of the status of the global resources RES available in the environment. However, given the decentralized setting, the status of the system has to be represented in a distributed way by considering the set VAR^i of status variables associated to each agent i. As usual in approaches based on Discrete Event Systems (DESs), each variable $v \in VAR^i$ assumes values in a predefined and finite domain $Dom(v)$.

The set of status variables VAR^i is partitioned into two subsets: END^i and ENV^i. END^i includes the endogenous variables which characterize the specific agent i (and therefore there is no direct relation between the endogenous variables of two agents i and j); ENV^i includes all the variables concerning the environment (e.g., the status of a resource, or the position of an object). Note that, because of the partitioning, each agent i maintains a private copy of the variables in ENV^i; 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. The consistency among the private copies is guaranteed by the existence of precedence links in RE: the status of a resource is known only by the agent that holds it, for all the other agents the status of the resource is *not-available*.

As noted earlier, the relinquishment/acquisition of a resource happens through the exchange of messages between the agent who releases the resource and the agent who gets the resource (this is performed via the send and receive actions associated to the precedence link).

Action models. The model of an action a takes into account not only the nominal effects of the action, but also the non deterministic effects of exogenous events affecting the action execution. Formally, an action model is the tuple $\langle pre(a), eff(a), event(a), \Delta(a)\rangle$ where $pre(a)$ and $eff(a)$ are subsets of VAR^i, representing the variables over which the preconditions and the effects are defined, respectively; $event(a)$ is a subset of exogenous events in $\mathcal{E} \cup \{\epsilon\}$ that may occur during the execution of a. Finally, $\Delta(a)$ is a transition relation modeling how the status of agent i changes after the execution of a. In particular, the state transition $\langle s_l, \epsilon, s_{t+1}\rangle \in \Delta(a)$ models the case of the nominal behavior: ϵ denotes the case where there is no occurrence of any exogenous event, while s_l and s_{l+1} represent two agent states (i.e., two complete assignments of values to the status variables in VAR^i) at the steps l and $l + 1$, respectively. The state transition $\langle s_l, e, s_{l+1}\rangle$ (with $e \in \mathcal{E}$), denotes the occurrence of the exogenous event e; note that it is easy to model, within the transition relation $\Delta(a)$, the non deterministic effects of e. Since in some domains it could be impossible (or too costly) to provide a complete model of the effects of a exogenous event, we extend the domain $Dom(v)$ of each variable $v \in VAR^i$ by including the value *unknown*; when a variable assume the value *unknown* in s_{l+1} the actual value of v is no more predictable.

Running example. Let us assume that, in our blocks world example, the agent `A2` loses the object `Om` while `A2` is moving from `L2` to `L3` (action a_7 of Fig. 1). To take into account this possibility, the transition relation of the *carry* action includes, among others, a state transition describing the effects of the exogenous event *lose-object* on the status of agent `A2`. Intuitively, the *lose-object* event changes the value of the variable `A2`.*carrying* from `Om` to *empty*; at the same time, the variable `Om`.*position* (representing the position of the object) changes from *on-board*-`A2` to *unknown*; in this case, in fact, it is unrealistic to have a precise model for the *lose-object*, and some of its effects cannot be anticipated.

4 Dealing with Ambiguity in MAP Monitoring

In the distributed framework proposed in this paper, each agent is responsible for monitoring the actions it executes. Intuitively, the monitoring task has to *keep track* of the agent status while the agent is executing the actions in its local plan, and to*detect* as soon as possible anomalous discrepancies between the expected nominal behavior of the agent and the observed one. To meet the first objective, each agent i maintains a "history", namely a *trajectory*, representing the sequence of state transitions occurred during the execution of a plan segment. To meet the second objective, the agent i must be able to determine the outcome of the actions it has executed; in fact, an anomalous execution manifests itself when the nominal effects of an action have not been achieved.

The monitoring task is made complex not only because of the non determinism of the action model, but also because of the very partial observability of the system, which impacts the monitoring process in two ways: first, the trajectory of agent i cannot be precisely estimated, (and therefore a set of alternatives, called *trajectory-set*, must be maintained); second, the agent i must be able to deal with ambiguous action outcomes: i could not be able to determine whether the nominal effects of an action have been achieved.

Agent Trajectory and Trajectory-set. An *agent trajectory*, denoted as $tr^i(0,l)$, is defined over a segment $[a_0^i, \ldots, a_l^i]$ of the local plan P^i, and consists of an ordered sequence of agent states and exogenous events representing an evolution of the status of agent i consistent with the observations it has received so far, more formally:

Definition 1. *The agent trajectory $tr^i(0,l)$ over the plan segment $P^i[a_0, \ldots, a_l]$ is $tr^i(0,l) = \langle s_0, e_0, s_1, \ldots, e_l, s_{l+1} \rangle$, where:*

s_k $(k : 0..l+1)$ is the state of agent i at the k-th step such that $obs^i(k) \cup s_k \not\vdash \perp$.
e_h $(h : 0..l)$ is an event in $\mathcal{E} \cup \{\epsilon\}$ occurring during execution of action a_h, involved in the agent state transition from s_h to s_{h+1}.

As mentioned above, we do not assume that the available system observability guarantees to precisely determine the status of an agent after the execution of each action. As a consequence of this ambiguity, the structure that the agent i has to maintain is the *trajectory-set* $Tr^i[0..l]$, which includes all the possible agent trajectories $tr^i(0, l)$ consistent with the observations received during the execution of the plan segment $P^i[a_0, \ldots, a_l]$.

Albeit the trajectory-set $Tr^i[0, l]$ maintains the history in the interval $[0, l+1]$, it is sometimes useful to single out the agent *belief state* at a given step k :

Definition 2. *Given the consistent trajectory-set* $Tr^i[0..l]$*, the agent* belief state $\mathcal{B}^i_k$ *(*$k : 0..l+1$*) is the set of all the consistent agent states inferred at the k-th step. Formally,* $\mathcal{B}^i_k = \text{PROJECTION}_{s_k}(Tr^i[0..l])$*.*

Action Outcomes. Intuitively, the outcome of an action a is a synthetic piece of information which states whether the nominal effects of a have been achieved or not; of course, in the positive case the outcome of a is *succeeded*, the outcome of a is *failed* otherwise.

In the Relational framework we propose, the nominal effects of an action a result from the following expression:

$$nominalEff(a) = \text{PROJECTION}_{eff(a)}(\text{SELECT}_{e=\epsilon}\Delta(a)).$$

Namely, $nominalEff(a)$ is the complete assignment of values to the status variables in $eff(a)$, when no exogenous event occurs (i.e. $e = \epsilon$).

The main problem in assessing the outcome of an action is the inherent ambiguity both in the action model and in the trajectory-set. For instance, in order to assess the outcome action a^i_l, agent i needs to check whether the nominal effects of a^i_l are satisfied in the agent belief state $\mathcal{B}^i_{l+1}$. Unfortunately, given the partial system observability, the agent belief state $\mathcal{B}^i_{l+1}$ is in general ambiguous: it contains states where the nominal effects of a^i_l hold, and other states where they do not hold. The following, conservative definitions allows agent i to univocally determine the success or the failure of action a^i_l.

Definition 3. *The outcome of action* a^i_l *is*

succeeded *iff for each state* $s \in \mathcal{B}^i_{l+1}$*,* $s \vdash nominalEff(a^i_l)$*.*
failed *iff for each state* $s \in \mathcal{B}^i_{l+1}$*,* $s \cup nominalEff(a^i_l) \vdash \bot$

However, in all those cases where these two definitions are not applicable we adopt a weak commitment policy which allows the outcome of an action to be *pending*; the assessment of the outcome is postponed till the belief state $\mathcal{B}^i_{l+1}$ is sufficiently refined to conclude either the success or the failure. To this end, the agent i maintains a list pO^i of actions whose outcome has not been (yet) determined.

The incremental monitoring process. Let us assume that the action a^i_l requires as precondition the service q provided by action a^j_m (i.e., by another agent j). Thus, as soon as action a^j_m has been executed, agent j has to notify agent i about the result of such an action by sending a message $msg(a^j_m, a^i_l)$ to agent i. Such a message includes only the tuple $\langle \epsilon, q \rangle$ in case agent i can univocally detect the success of action a^j_m (i.e., no exogenous event occurred and the service q has been provided). In case the outcome of action a^j_m is pending (i.e., agent j is not sure about the achievement of service q), the message includes not only the nominal situation $\langle \epsilon, q\rangle$, but also the anomalous situation $\langle exo(j, a^j_m), \neg q\rangle$, where the service q is not achieved because of the occurrence of an exogenous event denoted as $exo(j, a^j_m)$.

Since the action a^j_m should provide the service q to action a^i_l, the agent i has to consume the message $msg(a^j_m, a^i_l)$ during the monitoring of action a^i_l; in particular:

Definition 4. $Tr^i[0, l] = \text{SELECT}_{obs^i(l)}[[Tr^i[0, l-1] \times msg(a^j_m, a^i_l)] \text{ JOIN } \Delta(a^i_l))]$

Intuitively, the Relational product $Tr^i[0, l-1] \times msg^i_l$ is the mechanism through which the agent i includes within its trajectory-set the info provided by agent j. The join operation appends each possible transition in $\Delta(a^i_l)$ at the end of each trajectory $tr^i(0, l-1)$ in $Tr^i[0, l-1]$ iff the agent state s_l (i.e., the last agent state in the agent trajectory included the info provided by the message) satisfies the preconditions of the action a^i_l. The selection operation prunes the estimations by removing all the predicted trajectories which are inconsistent with the observations received as a feedback for the execution of a^i_l. It is important to note that the selection operator has an impact on the whole trajectory-set: the agent trajectory $tr^i(0, l)$ is removed from $Tr^i[0, l]$ when the last agent state s_{l+1} is inconsistent with the observations. As we will discuss later, through this mechanism it is possible to refine the past history of the agent status, and possibly determine the outcome of some past actions.

However, the above mechanism is not complete: it is possible, in fact, that for some trajectories $tr^i(0, l-1) \in Tr^i[0, l-1]$, the last state s_l does not match any of the transitions in $\Delta(a^i_l)$; this happens when some variables in s_l mention the symbol *unknown* (i.e., that variables are no longer predictable). In order to be able to incrementally extend also these trajectories for which the model of the action is not directly applicable, the monitoring step makes use of a weak prediction model:

Definition 5. *Let* $tr^i(0, l-1)$ *be an agent trajectory in* $Tr^i[0, l-1]$ *such that* s_l *does not satisfy the preconditions for action* a^i_l*; the agent trajectory* $tr^i(0, l-1)$ *is extended by appending a new state transition* $\langle s_l, \epsilon, s_{l+1}\rangle$*, where for each variable* $v \in VAR^i$*:*
v assumes the value unknown *in* s_{l+1} *iff* $v \in \text{eff}(a^i_l)$
v assumes in s_{l+1} *the same value assigned in* s_l *otherwise*

The first condition states that the effects of the action a^i_l become no longer predictable; the second condition imposes the persistency for all those variables which are not included in the definition of the nominal effects of the action a^i_l.

Observe that also the predictions inferred by means of the weak model must be consistent with the observations; when a variable v is *unknown* in s_l, it assumes in s_{l+1} the observed value in $obs(a^i_l)$, that is, the value *unknown* matches with any possible observation.

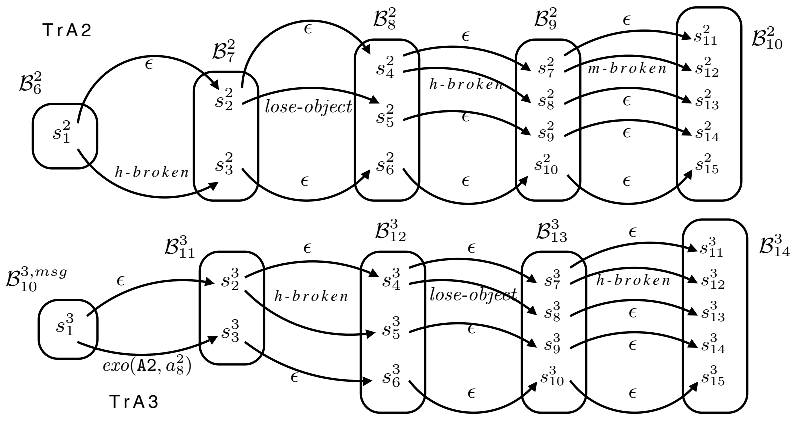

Fig. 2. The trajectory-sets computed by agents A2 and A3

Running example. Figure 2 shows two trajectory-sets, $TrA2$ and $TrA3$, built during the plan execution by agents A2 and A3 respectively; these trajectory-sets are consistent with the observations received by the agents so far. Note that each rectangle encloses all the agent states within a given belief state. For instance, the belief state $\mathcal{B}^2_9$ includes the states s^2_7, s^2_8, s^2_9, and s^2_{10}.

The trajectory-sets keep trace of the possible occurrence of exogenous events during the execution of the actions: for example, the exogenous event *lose* means that the agent can lose a block during a *carry* action; whereas *m-broken* and *h-broken* refer to a fault respectively in the mobility and handling functionality of the agent, the first fault affects the *move* action, the second affects the *load/unload* actions.

More important, since there exists a causal link between actions a^2_8 and $a_1 1^3$, the agent A2 sent a message to agent A3 about the service AT(Om,L3) (see Figure 1). However, action a^2_8 has a pending outcome, thereby the message received by A3 is ambiguous and maintains a reference to an exogenous event, which possibly occurred during the execution of a^2_8; this ambiguous message has been consumed by agent A3 and included in the trajectory-set $TrA3$ through the two transitions between the belief states $\mathcal{B}^{3,msg}_{10}$ and $\mathcal{B}^3_{11}$: the transition labeled with ϵ models the accomplishment of service AT(Om,L3), the transition labeled with $exo(\mathtt{A2}, a^2_8)$ models the occurrence of an exogenous event.

5 Cooperative Plan Monitoring and Diagnosis

In this section we discuss how the trajectory-set inferred by agent i can be refined by exploiting pieces of information provided by other team members. The refinement of the trajectory-set is an important step, which allows agent i to determine the outcome of some pending actions in pO^i. To reach this objective, one has to single out which pieces of information should be exchanged among the agents and when.

```
agent i
procedure PropagateSuccess($a^i_l$) {
  for each link $cl \in T^i_{in} | cl : a^j_m \xrightarrow{q} a^i_l$
    notify agent $j$:
      service $q$ in $cl : a^j_m \xrightarrow{q} a^i_l$ accomplished
}
```

```
agent j
procedure ExploitSuccess($q$, $cl$) {
  assert $q$ in $\mathcal{B}^j_{m+1}$ and prune $Tr^j$
  assert outcome($a^j_m$) = succeeded
  revaluate outcome for each action in $pO^j$
}
```

Fig. 3. The procedures for success propagation

In principle, the agents could exchange one another the observations they receive from the environment. However, this approach may suffer from two drawbacks: first, the amount of data the agents need to exchange could be too large; second, the data concerning the observations received by an agent could not be necessarily useful for another agent.

As discussed in the previous section, in our approach the agents exchange the results of a local interpretation process aimed at inferring the action outcome. In particular, the messages are sent not only when an agent detects the success or failure of an action, but also in presence of ambiguity (pending outcome). In the following of the section, we will show how the ambiguous results provided by agent j to agent i can be refined on the basis of the feedback provided by agent i, so that agent j can determine the outcome of some previously pending action.

First of all, let us consider the case when agent i detects the nominal outcome for action a^i_l on the basis of the observations $obs(a^i_l)$. In this case, the trajectories in $Tr^i[0..l]$ resulting from the monitoring step involve only ϵ transitions, and the agent i can conclude that the execution of all previous actions is nominal and all the services needed for action a^i_l have been provided; in this way it is possible to determine the outcome of some pending action in pO^i. Moreover, the nominal outcome of action a^i_l provides a positive feedback to agent j, in fact the following property assures that at least the nominal outcome of action a^j_m is detected.

Property 1. Let a^i_l be an action in A^i, and let a^j_m an action in A^j such that there exists a causal link $cl \in T^i_{in}$, $cl : a^j_m \xrightarrow{q} a^i_l$, and let us assume the outcome of action a^j_m is pending; if a^i_l has outcome *succeeded* then q has been certainly accomplished and a^j_m has outcome *succeeded.*

For this reason agent i invokes procedure **PropagateSuccess** (see Figure 3) to notify other agents about this outcome. Observe that the propagation is performed by considering only the subset of agents which provide action a^i_l with some service q (see the incoming causal links in T^i_{in}).

The message about the accomplished service q is exploited by agent j, by invoking procedure **ExploitSuccess**, whenever the agent j is not sure to have provided q. To consume this message, the agent j asserts the atom q within its trajectory-set Tr^j; more precisely, since the atom q refers to the effects of action a^j_m, q must be asserted in the agent belief state inferred after the execution of a^j_m, namely $\mathcal{B}^j_{m+1}$. It is worth noting that as a side effect, the trajectory-set Tr^j can be refined by pruning off all those agent trajectories which are not consistent with q. Therefore, after this first step, the agent j reconsiders each pending action

```
agent i
procedure BackPropagateFailure ($a_l^i$){
  infer local diagnosis $D^i$
  let exist $cl \in T_{in}^i \mid cl : a_m^j \xrightarrow{q} a_l^i$
  if the only explanation in $D^i$ refers to $exo(j, a_m^j)$
    notify agent $j$ service $q$ has not been provided
}

agent j
procedure ExploitBackPropFailure($q$, $cl$){
  assert $\neg q$ in $\mathcal{B}_{m+1}^j$ (pruning of $Tr^j$)
  assert outcome($a_m^j$) $=$ failed
  revaluate outcome for each action in $pO^j$
  infer local diagnosis $D^j$
}
```

Fig. 4. The procedures for back propagation of an action failure

in pO^j, as it is possible that the trajectory-set Tr^j has been sufficiently refined to determine the outcome of some of them.

In case agent i detects the failure of action a_l^i, a local diagnosis process is immediately activated to provide some possible explanations for that failure. A local diagnosis can be inferred directly from the trajectory-set as follows.

Definition 6. *Given the failure of action* a_l^i*, and the trajectory-set* $Tr^i[0..l]$*, the* local diagnosis *for that failure is* $D^i=$ PROJECTION $_{e_0,\ldots,e_l} Tr^i[0..l]$.

In other words, the local diagnosis for the failure of a_l^i is a set of sequences of events, where each sequence *seq* has the form $\langle e_0, e_1, \ldots, e_l \rangle$ and represents a possible explanation. Each event e_k $(k : 0..l)$ is in $\mathcal{E} \cup \{\epsilon\}$, however, since a not nominal outcome has been detected, in each sequence *seq* at least one anomalous event must be occurred.

It is important to note that some of the explanations included in the local diagnosis could refer to anomalous events concerning services provided by other agents; that is, it is possible to explain the failure of action a_l^i as an indirect consequence of the failure of some actions performed by other agents.

In particular, when agent i is able to explain its local failure just as a consequence of a failure in the local plan by agent j, it invokes the procedure **Back-PropagateFailure** (see Figure 4), to notify agent j that service q has not been provided. Whenever agent j receives such a message activate procedure **Exploit-BackPropFailure**: the local trajectory-set of agent j is refined by asserting $\neg q$ in $\mathcal{B}_{m+1}^j$; after this step it is therefore possible for agent j to determine whether other actions, besides a_m^j, are failed, and hence the outcome for each pending action in pO^i is evaluated again. Finally, agent j infers the local diagnosis D^j according to definition 6.

Running example. Let us consider again the trajectory-sets in figure 2, and assume that after the execution of action a_9, agent `A2` receives the message *"Position equals L2"*. This piece of information is used to prune the trajectory-set of `A2`, which is able to conclude that action a_9 has outcome *succeeded* (the agent knows that it has reached position *L2* as exepected). However, `A2` does not know whether the object `Om` has been delivered to agent `A3` or not, thus its set of pending outcomes is $pO^{\mathtt{A2}}=\{a_6, a_7, a_8\}$.

Now, let us assume that the set of pending outcomes for agent `A3` is $pO^{\mathtt{A3}}= \{a_{11}, a_{12}, a_{13}\}$, but after the execution of action a_{13} the agent `A3` receives the observation *"Object Om on top of object Oh"*, also in this case the observation is used to prune the trajectory-set of A3, in particular the result of the pruning consists in removing all the anomalous trajectories, and as a consequence A3 determines the nominal outcome for each action in $pO^{\mathtt{A3}}$. Observe that, as soon as agent A3 determines the nominal completion of action $a_1 1$, depending on services provided by A2, it notifies A2 that those services have been provided. In fact, the nominal outcome of action a_{11} implies also a nominal outcome for action a_8, which in turn implies the successful completion also for action a_6 and a_7.

6 Discussion and Conclusion

In recent years increasing attention has been devoted to plan execution and in particular to plan diagnosis ([1,6]). In fact, the early detection of an action failure, and of a possible explanation of its causes, are essential to start a recovery step (see e.g. [4]). The framework by Roos et al. [2] has many similarities with the framework we propose as it considers a precise notion of multi-agent plan, where actions are atomic and concurrently performed by a team of agents. However, Roos et al. discuss a centralized approach: the diagnostic problem takes into account all the agents in the team and all the available observations at a given time. Moreover, their action models consider just the nominal action behavior, while any abnormal behavior is *unknown*. On the contrary, the framework we have discussed is distributed: each agent is responsible both for executing actions and for diagnosing them. In addition to that, we can model the nominal as well as the anomalous behavior of an action; in particular, the anomalous behavior may be non deterministic or even abstracted by *unknown*.

It is important to note that the complexity of the plan diagnosis task strongly depends on the amount of observations available to the agents. In previous works ([3]) we have described methods able to detect and diagnose action failures when each agent has sufficient information for certainly detecting the outcome at least of each action providing other agents with services.

In the present paper we have considered a very challenging scenario where the degree of observability is so low, that an agent may not determine the outcome of an action even when that action provides services to other agents. A demanding consequence of such a scenario is that the agents must be able to deal with ambiguous action outcomes and need to communicate one another to refine as far as possible their beliefs. To face this problem we presented the weak-commitment policy, which allows *pending* outcomes to be turned into success or failure through the exchange of messages among agents. Notice that messages are not raw data (such as low-level observations), but are action outcomes. The advantage of this solution is twofold: first, agents reduce the amount of data to be communicated; second, agents exchange much more informative data since action outcomes are the results of (local) interpretation processes.

A preliminary set of experiments is currently carried on for testing the approach, which has been implemented by extending the software prototypes used in [3], and by exploiting the symbolic formalism of the Ordered Binary Decision Diagrams for compactly encoding both action models and trajectories.

References

1. Kalech, M., Kaminka, G.A.: On the design of coordination diagnosis algorithms for teams of situated agents. Artificial Intelligence 171(8-9), 491–513 (2007)
2. Roos, N., Witteveen, C.: Models and methods for plan diagnosis. Journal of Autonomous Agent and MAS 19(1), 30–52 (2009)
3. Micalizio, R., Torasso, P.: Monitoring the execution of a multi-agent plan:dealing with partial observability. In: Proc. of ECAI 2008, pp. 408–412 (2008)
4. Micalizio, R.: A distributed control loop for autonomous recovery in a multi-agent plan. In: Proc. of the 21st IJCAI 2009 (to appear, 2009)
5. Cox, J.S., Durfee, E.H., Bartold, T.: A distributed framework for solving the multiagent plan coordination problem. In: Proc. AAMAS 2005, pp. 821–827 (2005)
6. Horling, B., Benyo, B., Lesser, V.: Using self-diagnosis to adapt organizational structures. In: Proc. Int. Conf. on Autonomous Agents (ICAA 2001), pp. 529–536 (2001)