Auton Agent Multi-Agent Syst (2009) 18:267–294
DOI 10.1007/s10458-008-9045-x

Primary and secondary diagnosis of multi-agent plan execution

Femke de Jonge · Nico Roos · Cees Witteveen

Published online: 12 April 2008
The Author(s) 2008

Abstract Diagnosis of plan failures is an important subject in both single- and multi-agent planning. Plan diagnosis can be used to deal with plan failures in three ways: (i) to provide information necessary for the adjustment of the current plan or for the development of a new plan, (ii) to point out which equipment and/or agents should be repaired or adjusted to avoid further violation of the plan execution, and (iii) to identify the agents responsible for plan-execution failures. We introduce two general types of plan diagnosis: *primary plan diagnosis* identifying the incorrect or failed execution of actions, and *secondary plan diagnosis* that identifies the underlying causes of the faulty actions. Furthermore, three special cases of secondary plan diagnosis are distinguished, namely *agent diagnosis*, *equipment diagnosis* and *environment diagnosis*.

Keywords Diagnosis · Planning · Multi-agent systems

1 Introduction

In multi-agent planning research there is a tendency to deal with plans that become larger, more detailed and more complex. As complexity grows, the vulnerability of plans for failures will grow correspondingly. Taking appropriate measures to plan failures requires the ability to detect the occurrence of failures as well as to determine their causes. Therefore, we

F. de Jonge · N. Roos (✉)
Department of Computer Science, Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands
e-mail: roos@micc.unimaas.nl

F. de Jonge
e-mail: f.dejonge@micc.unimaas.nl

C. Witteveen
Faculty EEMCS, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
e-mail: C.Witteveen@tudelft.nl

consider diagnosis as an integral part of the capabilities of agents in single- and multi-agent systems.

To illustrate the relevance of plan diagnosis, consider a simple example in which your luggage did not arrive at the destination when flying from New York to Amsterdam. After *detecting this fault*, i.e., your luggage did not appear on the arrival belt, you know that the plan of transporting your luggage did not lead to the desired result. Clearly, the normal plan execution has failed and we wish to apply diagnosis to find out why. Given the transport plan of your luggage, consisting of actions such as sorting the luggage after check-in, loading it on the airplane, unloading the luggage, and so on, we apply diagnosis to identify which of these actions have failed. For instance, sorting the luggage to the right airplane. We will denote this type of diagnosis as *primary plan diagnosis*. Primary plan diagnosis focuses on a set of fault behaviors of *actions* that explain the differences between the expected and the observed plan execution.

Often, however, it is more interesting to determine the *causes behind* such faulty action executions. In our example, the failure of the action 'sorting of luggage' may be caused by a malfunction of the equipment used. For instance, the component that reads the label attached to the luggage may make errors. We will denote the diagnosis of these underlying causes as *secondary plan diagnosis*. Secondary diagnosis can be viewed as a diagnosis of the primary diagnosis. It informs us for instances about a malfunctioning of the *equipment* that caused the faulty behavior of an action performed by it. Beside malfunctioning equipment, secondary diagnosis may also identify the *agents* that did not execute an action correctly and changes in the *environment* that where not anticipated at the time the plan was made. For instance, loading the luggage onto an airplane may fail for a piece of luggage because the personnel forgot to load that piece. Unexpected environment changes (such as the weather) may also influence the outcome of an action. For instance, very strong turbulence during a flight may cause injuries among passengers and may damage the cargo.

As a special type of secondary diagnosis, we are also able to determine the agents *responsible* for the failed execution of some actions. In our luggage transportation example, the maintenance agent might be responsible for the malfunctioning sensor that reads the labels of luggage.

In our opinion, diagnosis in general, and secondary diagnosis in particular, enables the agents involved to make specific adjustments to the system or the plan as to manage current plan-execution failures and to avoid new plan-execution failures. These adjustments can be categorized with regard to their benefits to the general system. Primary diagnosis can contribute to plan repair by identifying the failed action and how it failed. Secondary diagnosis contributes to plan repair by identifying the broken equipment and the misbehaving agents. In particular, information about misbehaving agents may help to adjust the agents, thereby contributing to a better plan execution. Finally, by indicating the agents responsible for plan failures, secondary diagnosis also becomes a valuable tool in evaluating the system and might be used to divide costs of repairs and/or changes in the plan amongst the agents.

1.1 Aim, approach and results

In some previous papers [30,24] we have introduced a simple formal framework for plan diagnosis. The framework is based on classical Model-Based Diagnosis (MBD). We have shown that modeling the world using a set of state variables enables us to view (i) actions as components, (ii) variables who's values are set by one action and are inputs of the next action as connections between component, and (iii) the whole plan as a system.

In this framework we restricted diagnosis to primary diagnosis of failing actions in a plan, distinguishing only two health modes for actions: a normal (*nor*) and an abnormal (*ab*) mode. Using the framework we were able to point out some computational differences between two types of preferred plan diagnoses: minimum diagnoses and minimal maximum-informative diagnoses (mini-maxi diagnoses). While minimum diagnoses turned out to be difficult to compute, mini-maxi diagnoses were computable in polynomial time.

We did not, however, investigate a more refined set of fault modes and we could not identify the *causes* of failing actions as e.g., executing agents, equipment used to execute an action, or unforeseen changes of the environment. The purpose of this paper is to extend this existing framework in several respects.

First of all, we would like to have a common framework for plan diagnosis to deal with both primary and secondary diagnosis, where agents, equipment and environment could be identified as possible sources for plan execution errors. This necessitates us to introduce health modes for agents, equipment and environment, too. Equipment has health modes indicating the normal behavior (*nor*), the general unknown but abnormal behavior (*ab*), and equipment-specific health modes. The label reader in the above example for instance may have a health mode indicating that it confuses 'AMS' (Amsterdam) with 'AMM' (Amman). An agent executing a part of a plan can be viewed as just a special type of equipment. Hence, besides the health modes *nor* and *ab*, an agent may have agent-specific health modes for hardware, software, sensor and actor failures.

Remark 1 Beside viewing agents as a special kind of equipment, we may also view agents as intentional systems (beings). From this perspective, agents may fail to execute an action correctly because of incorrect beliefs or intentions. In such an agent, we may view the agent's belief states as health modes.[1] The diagnostic model presented in the following sections is also able to identify these causes of execution failures. Note that identifying belief states of agents is related to *social diagnosis* [16,17]. However, social diagnosis does not deal with plans consisting of a partially ordered set of actions.

We also would like to allow health modes for environment objects. Although the purpose of executing a plan is to change the environment in such a way that after the execution the state of the environment satisfies some goals, the environment may also influence the execution of the plan in unintended ways. For instance very strong turbulence causing damage to fragile cargo. Therefore, diagnosis should be able to identify unforeseen environment changes influencing the execution of an action. Finally, we also allow for more than two health modes for actions. Besides the health modes *nor*, denoting that the action is executed as expected, and *ab*, denoting that the action execution fails, we allow for several *action-specific* health modes, such as *routing error: AMS* $\rightarrow$ *AMM* for luggage sorting action.

The above extensions needed to deal with primary and secondary diagnosis will cause a major change of the existing plan diagnosis framework.

First of all, we introduce a uniform representation of actions, agents equipment and the environment using a set of *state variables*. To be able to distinguish different types of diagnoses such as primary diagnosis and three types of secondary diagnosis, namely: agent, equipment and environment diagnosis, we partition these variables into *action*, *agent*, *equipment* and *environment* variables.

[1] However, in general we cannot state in advance which beliefs are normal modes and which are faults modes since this may depend on the context.

Secondly, we introduce *events* to describe the unforeseen (unplanned) state changes of variables. Variables may therefore be viewed a discrete event systems [5] with respect to the unforeseen state changes.[2]

Finally, we define diagnosis as a set of events changing the states of variables. We make a distinction between two types of diagnosis, namely primary and secondary diagnosis where the latter gives the underlying causes of the former. Secondary diagnosis is further divided into agent, equipment and environment diagnosis. Moreover as a special form of secondary diagnosis, we discuss the identification of agents responsible for failures. These agents need not be the agents executing the plan.

1.2 Organization

The remainder of the paper is organized as follows. In the next section, Sect. 2, we present some related work and discuss some related approaches to plan diagnosis. Then, in Sect. 3, we introduce a general framework for plan-diagnosis that enables the identification of different causes of plan execution failures as will be described in Sect. 4. In Sect. 5 we illustrate the aspects of our diagnostic model using a large example. Being able to efficiently identify diagnoses is an important issue. Section 6, discussed preference criteria on diagnoses as well as some complexity results with respect to the identification of preferred diagnoses. Section 7 concludes the paper.

To place our approach in perspective, before introducing the plan and diagnostic model, we first discuss some related work.

2 Related research

Our approach to plan diagnosis can be viewed as an extension of the Model-Based Diagnosis (MBD) to the planning domain. We therefore start with a brief overview of MBD and its connection to plans.

2.1 Model-based diagnosis

Classical Model-Based Diagnosis (MBD) assumes a set of connected components usually with specific inputs and outputs. A model of the system is used to predict its expected behavior. Discrepancies between the observed and the predicted behavior are subsequently used to determine a diagnosis. Two forms of diagnosis can be distinguished; abductive [6] and consistency-based diagnosis [23,11]. To apply abductive diagnosis we must be able to predict the values of all the observed system outputs using a description of the system. Consistency-based diagnosis is a weaker notion of diagnosis. The predictions made using a consistency-based diagnosis only need to be consistent with the observed values of the observed system outputs. Hence an abductive diagnosis is more restrictive but also requires more information about the system behavior [22]. Especially, one needs to know the behavior of malfunctioning components. Note that if the later knowledge is known, consistency-based diagnosis identifies the same set of diagnosis as abductive diagnosis [10].

We will show that a plan consisting of a partially ordered set of actions can be viewed as a system of connected components. This view enables us to predict the expected behavior of a plan, observe discrepancies between the predicted and the expected behavior and apply

[2] Note that the execution of actions will not be described in terms of events changing the state of a variable.

consistency-based diagnosis to identify causes. We do not consider abductive diagnosis since we need not know the behavior of failing or incorrectly executed actions of a plan.

2.2 (Multi-)agent-based approaches

Similar to our use of MBD as a starting point of plan diagnosis, Birnbaum et al. [2] apply MBD to *planning agents* relating health modes of agents to *outcomes* of their planning activities, but not taking into account faults that can be attributed to actions occurring in a plan as a separate source of errors.

de Jonge et al. [8,9] present an approach that directly applies MBD to plan execution. Here, the authors focus on agents each having an individual plan, and on the conflicts that may arise between these plans (e.g., if they require the same resource). Actions are viewed as processed, which are modeled as Discrete Event Systems (DES). Diagnosis is applied to determine those factors that are accountable for *future* conflicts. The authors, however, do not take into account dependencies between health modes of actions and do not consider agents that collaborate to execute a common plan.

Micalizio and Torasso [19] introduce a distributed monitoring approach for multi-agent plan execution which is similar to the work of de Jonge et al. They also view a plan's actions as processes and they also model actions as DESs. Their approach, however, differs from the work of [8,9] in that they explicitly incorporate the belief state of the executing agent in the description of actions. Moreover, they focus on monitoring and diagnosis of the agents' belief states.

In [20] the authors apply this approach to execution of a multi-agent plan and extend the approach with repair of faults occurring during the execution the plan. They view a plan as a partially order set of actions where each action provides a service required by following actions. The actions are again modeled as DESs. When an agent detects an action failure, it will first try to repair its own sub-plan. If this local recovery fails, the team to with the agent belongs will try to repair the plan.

Kalech and Kaminka [17,18] apply *social diagnosis* in order to find the cause of an anomalous plan execution by a team of agents. They consider hierarchical plans consisting of so-called *behaviors*. Such plans do not prescribe a (partial) execution order on a set of actions. Instead, based on an agent's observations, beliefs and role, the plan prescribes for each agent the appropriate behavior to be executed. Each behavior in turn may consist of primitive actions to be executed, or of a set of other behaviors to choose from. Social diagnosis then addresses the issue of determining what went wrong in the joint execution of such a plan by identifying the disagreeing agents and the causes for their selection of incompatible behaviors (e.g., belief disagreement, communication errors) based on monitoring the agents' plan execution.

Carver and Lesser [4] and Horling et al. [15] also apply diagnosis to (multi-agent) plans. Their research concentrates on the use of a *causal model* that can help an agent to refine its initial diagnosis of a failing *component* (called a *task*) of a plan. As a consequence of using such a causal model, the agent would be able to generate a new, situation-specific plan that is better suited to pursue its goal. Diagnosis is based on observations of a component without taking into account the consequences of failures of such a component w.r.t. the remaining plan.

In previous work [24,26], the authors show how classical MBD can be applied to identify abnormally executed actions in a plan consisting of a partially ordered set of actions. They proposed to describe the state of the world by a set of variables and the actions as functions that modify some of the state variables. This description of the world and the actions together with partial observations at different time points, makes it possible to apply classical MBD by viewing (i) actions as components of a system, and (ii) variables set by one action and required as inputs by a following action as connections between components. Unlike other approaches, this approach does not monitor the execution of all plan steps and does not model plan steps as processes. Since we do not monitor the execution of plan steps, we only monitor the effects of the plan execution at some time point, there is nothing to gain by modeling plan steps as processes. We can, however, gain something from modeling the specific health state of actions, agents and equipment. Below we present an adapted and extended version of our formalization of plan diagnosis. This formalization enables the handling of the health state of actions, agents and equipment in much the same way as the approaches of de Jonge et al. [8,9], Kalech and Kaminka [17,18], and Lesser et al. [4,15]. The work of Birnbaum et al. [2] is not covered by the proposed formalization since it focuses on the planning activity instead of on plan execution.

2.3 Discrete event systems

To model state changes that are not caused by the execution of actions, we propose to view state variables as *Discrete Event Systems* (DESs). This view enables us to describe unknown causes of state changes by unobserved *events* where the *transition function* places restriction on the state changes that may occur.

In general, Discrete Event Systems are a modeling method of real-world systems based on finite state machines (FSM) [5]. In a DES a finite set of states describes at some abstraction level the state of a real-world system. State changes are caused by events and a transition function specifies the changes triggered by the events. The events are usually observable control events. However, unobservable failure events mays also cause state changes. Diagnosis of a DES aims at identifying the unobserved failure events based on a trace of observable events [27]. Essentially, this is a form of abductive diagnosis. Note that the trace of observable events depends on the state of the system and the transition function. Therefore, a DES is sometimes viewed as a machine accepting a language of observable and unobservable events.

In order to create a DES modeling a system, one first models the behavior of *individual components* using DESs. The finite state machines described by the DESs modeling these components interact by exchanging events. Here, a state transition of a FSM may trigger new events causing state transitions of other FSMs. To make a diagnosis using the coupled DESs two main approaches exist. The oldest approach combines the FSMs modeling the components into a global FSM of the whole system [28]. Unfortunately, the number of states of the global finite state machine grows exponentially with the number of components, making it infeasible for modeling large systems. Therefore, in recent work, instead of creating a single FSM, methods for making diagnosis directly using the coupled DESs have been proposed in the literature [1,21].

Remark 2 Although we will propose the use of discrete event systems for describing state changes of variables caused by unknown events, we will not model plan execution using DESs. The reason is that an action in a plan is a process that changes the values of variables describing the state of the world. Since actions change the values of variables, they must be described by events. Of course we could avoid the combinatorial explosion associated with creating one global DES by introducing a DES for each variable (as we will do in this paper). However, describing actions as events implies ignoring the processing time of actions. Although, in this paper, we consider unit processing times for actions in order to simplify the description of our model, there is no principal objection against using non-unit processing times of actions. The use of events however to represent actions with non-unit processing times, which may also depend on the agents executing the actions, the equipment used the environment and the way actions may fail, is quite problematic.

Another reason why we do not model a plan using a DES is because diagnosis of DESs is essentially abductive diagnosis. This implies that we need to know all the different ways in which executing an action may fail. In general, this knowledge is not available.

3 Plans as systems

We consider plan based diagnosis as an extension of model-based diagnosis where the model is a description of a plan instead of a physical system. In this view the execution of an action is a component changing the state of the world. Therefore, we will introduce a state-based description of the world. The state-based description is changed by the execution of actions and by disruptions.

To realize plan diagnosis, we adapt and extend the representation of plans presented in [30,24]. This representation, in which the state of the world is described by a set of variables, has been chosen because it is more suited to plan diagnosis than the traditional plan representations such as [3,12,13,29] and it also enables us to extend the classical Model-Based Diagnosis (MBD) approach to the planning domain. Moreover, we wish to study plan diagnosis independent of any planning approach that is used to generate a plan or that will be used to repair a plan.

We now introduce the building blocks of our plan diagnosis approach. Since our representation differs from the traditional plan representation, we will relate our representation to STRIPS.

3.1 Actions, plan operators and plan steps

In the preceding sections we used the term ‘actions’ in a rather informal way. From now on we will distinguish *plan operators* $\mathcal{O}$ and *plan steps* $\mathcal{S}$, which are both covered by the term ‘actions’.

A *plan operator* $o \in \mathcal{O}$ refers to a general description of an action independently of how it is used in a plan. In contrast, a *plan step* $s \in \mathcal{S}$ is an instantiation of a *plan operator* to be used at a specific point in a plan. One plan operator $o \in \mathcal{O}$ may therefore have several instantiations $\{s_1, \dots, s_n\} = inst(o)$ that are used in the same plan. A plan operator for transporting items may, for instance, be instantiated resulting in a plan step for the transport of a specific container from Rotterdam to New York by ship, and also be instantiated resulting in a plan step for the transport of a passager’s luggage from Washington to Amsterdam by plane.

3.2 States and partial states

In [30], it was shown that by describing the state of the world using objects or variables instead of propositions, it becomes possible to apply classical MBD to plan execution. Here, we will take this approach one step further by also introducing variables for agents executing the plan, the equipment used by the agents and for the plan steps themselves. We explicitly introduce these variables in order to enable the identification of other health modes of steps beside *normal* and *abnormal*, and the identification of underlying causes of failing plan steps such as misbehaving agents and malfunctioning equipment. Hence, we assume a finite set of variables $\mathcal{V}$ that will be used to describe the plan, the agents, the equipment and the environment, respectively. The variables $\mathcal{V}$ are partitioned into four classes or types: *plan steps* $\mathcal{S}$, *agents* $\mathcal{A}$, *equipment* $\mathcal{E}$ and *environment variables* $\mathcal{N}$. Each variable in $v \in \mathcal{V}$ is assumed to have a domain D_v of values. The *state* of the set of variables $\mathcal{V} = \{v_1, \ldots, v_n\}$ at some time point is described by a tuple $\sigma \in D_{v_1} \times \cdots \times D_{v_n}$ of values. In particular, four projections $\sigma^{\mathcal{S}}$, $\sigma^{\mathcal{A}}$, $\sigma^{\mathcal{E}}$ and $\sigma^{\mathcal{N}}$ of σ are used to denote the state of the plan step variables $\mathcal{S}$, the agent variables $\mathcal{A}$, the equipment variables $\mathcal{E}$ and the environment variables $\mathcal{N}$.

The state $\sigma^{\mathcal{N}}$ of the set of environment variables $\mathcal{N}$ describes the state of the agents' environment at some point in time. For instance, these state descriptions can represent the location of an airplane or the availability of a gate.

The states $\sigma^{\mathcal{S}}$ and $\sigma^{\mathcal{E}}$ of plan step and equipment variables, describe the health of plan steps and equipment, respectively. Their domains consist of the health modes of the plan steps and equipment. We assume that each of these domains contains at least (i) the value *nor* to denote that the plan steps and equipment behave normally, and (ii) the general fault mode *ab* to denote that the plan steps and equipment behave in an unknown and possibly abnormal way. Moreover, the domains may contain several more specific fault modes. For instance, different types of routing errors for the 'sort luggage' action in the example in the Introduction. Note that beside a variable describing the health state of a piece of equipment, a separate environment variable must be introduced to describe for instance the location of the piece of equipment. The equipment variables are reserved for describing the health modes of the equipment.

If an agent is viewed as a special type of equipment, then state $\sigma^{\mathcal{A}}$ either describes the agents' health mode in terms of normal operation *nor*, general malfunctioning *ab*, and other health modes denoting hardware, software, sensor and actor failures. If, however, an agent is viewed as an intentional system, the health modes may describe the agent's beliefs that determine the agent's execution a planed plan step. Incorrect beliefs may result in an incorrect execution of a plan step. The actual choice depends on the application domain. If necessary, two variables representing both aspects may be used for each agent.

It will not always be possible to give a complete state description. Therefore, we introduce a *partial state* as an element $\pi \in D_{v_{i_1}} \times D_{v_{i_2}} \times \cdots \times D_{v_{i_k}}$, where $1 \leq k \leq n$ and $1 \leq i_1 < \cdots < i_k \leq |\mathcal{V}|$. We use $V(\pi)$ to denote the set of variables $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\} \subseteq \mathcal{V}$ specified in such a state π. The value of a variable $v \in V(\pi)$ in π will be denoted by $\pi(v)$. The value of an object $v \in \mathcal{V}$ not occurring in π is said to be *unknown* (or unpredictable) in π, denoted by $\perp$. Including $\perp$ in every value domain D_i allows us to consider every partial state π as an element of $D_1 \times D_2 \times \cdots \times D_{|\mathcal{V}|}$.

Partial states can be ordered with respect to their information content: given values d and d', we say that $d \leq d'$ holds iff $d = \perp$ or $d = d'$. The containment relation $\sqsubseteq$ between partial states is the point-wise extension of $\leq$: π is said to be contained in π', denoted by $\pi \sqsubseteq \pi'$, iff $\forall v \in \mathcal{V}\ [\pi(v) \leq \pi'(v)]$.

Given a subset of variables $V \subseteq \mathcal{V}$, two partial states π, π' are said to be V*-equivalent*, denoted by $\pi =_V \pi'$, if for every $v \in V, \pi(v) = \pi'(v)$. We define the partial state π restricted to a given set V, denoted by $\pi\lceil V$, as the state $\pi' \sqsubseteq \pi$ such that $V(\pi') = V \cap V(\pi)$.

An important notion in plan diagnosis is the notion of *compatibility* between partial states. Intuitively, two states π and π' are said to be compatible if they could characterize the same state of the world, that is, there exists a state σ such that $\pi \sqsubseteq \sigma$ and $\pi' \sqsubseteq \sigma$. Equivalently, this compatibility relation can also be expressed without making a reference to such a state σ since the existence of such a state σ clearly implies that for every $v \in Var$, either $\pi(v) = \pi'(v)$ or at least one of the values $\pi(v)$ and $\pi'(v)$ is undefined:

Definition 1 (*compatibility relation*) Two partial states π and π' are said to be compatible, denoted by $\pi \approx \pi'$, iff

$$\forall v \in \mathcal{V}[\ \pi(v) \leq \pi'(v)\ or\ \pi'(v) \leq \pi(v)\ ].$$

3.3 Goals

An (elementary) goal g of an agent specifies a set of states an agent wants to bring about using a plan. Here, we specify each such a goal g as a constraint, that is, a relation over some product $D_{i_1} \times \cdots \times D_{i_k}$ of domains. We say that a goal g is satisfied by a partial state π, denoted by $\pi \models g$, if the relation g contains some tuple (partial state) $(d_{i_1}, d_{i_2}, \ldots d_{i_k})$ such that $(d_{i_1}, d_{i_2}, \ldots d_{i_k}) \sqsubseteq \pi$. We assume each agent a to have a set G_a of such elementary goals $g \in G_a$. We use $\pi \models G_a$ to denote that all goals in G_a hold in π, i.e. for all $g \in G_a$, $\pi \models g$.

3.4 Plan step execution

The execution of a specific plan step $s \in \mathcal{S}$ may not only change the state of environment variables $\mathcal{N}$, but also agent variables $\mathcal{A}$ and equipment variables $\mathcal{E}$. We describe such changes induced by a specific plans step $s \in \mathcal{S}$ by a function f^o:

$$f^o : D_s \times (D_{a_1} \times \cdots \times D_{a_i}) \times (D_{e_1} \times \cdots \times D_{e_j}) \times (D_{n_1} \times \cdots \times D_{n_k}) \rightarrow$$
$$(D_{a'_1} \times \cdots D_{a'_u}) \times (D_{e'_1} \times \cdots \times D_{e'_v}) \times (D_{n'_1} \times \cdots \times D_{n'_w})$$

where

- o is the *plan operator* of which the plan step s is an instance, that is, $s \in inst(o)$,
- $a_1, \ldots, a_i \in \mathcal{A}$ are the execution agents,
- $e_1, \ldots, e_j \in \mathcal{E}$ are the variables describing the health modes of equipment required,
- $n_1, \ldots, n_k \in \mathcal{N}$ are the environment variables required, and
- $\{a'_1, \ldots, a'_u, e'_1, \ldots, e'_v, n'_1, \ldots, n'_w\}$ are agent, equipment and environment variables, respectively, that are changed by the execution of plan step s.

Each application of the function f^o associated with the plan operator o is completely determined by a plan step $s \in inst(o)$. Therefore, we will use the plan step s to denote the binding of the variables to the *domain parameters* of the function f^o by $dom_{\mathcal{V}}(s) = \{s, a_1, \ldots, a_i, e_1, \ldots, e_j, n_1, \ldots, n_k\}$. Likewise we denote the binding to the *range parameters* by $ran_{\mathcal{V}}(s) = \{a'_1, \ldots, a'_u, e'_1, \ldots, e'_v, n'_1, \ldots, n'_w\}$.

Remark 3 Note that a plan step is the only variable that cannot be changed by the execution of a plan step. A plan step is executed only once and its health mode determines how it is executed. We allow that the values of all other variables can be changed by the execution of a plan step. In case of equipment, such a change may indicate effect of a maintenance or a repair activity, but also the effect of damaging equipment by operating it outside the operation parameters. The same holds for agents if agents are viewed as a special kind of equipment.[3] Also note that the set of variables $ran_{\mathcal{V}}(s)$ bound to the range of f^o with $s \in inst(o)$ may differ from the variables $dom_{\mathcal{V}}(s)$ bound to the domain of f^o. This property makes it possible to set the value of a variable independently of its previous value. Finally, note that most planning formalisms, such as STRIPS, can be modeled using our formalism for describing plans. The relation with STRIPS will be described in paragraph "The relation with STRIPS".

[3] If, however, agents are viewed as intentional systems, state changes may indicate belief updates.

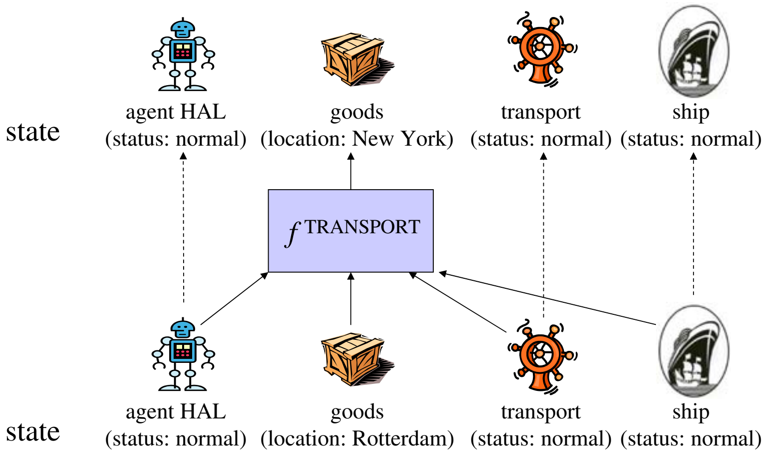
state
agent HAL (status: normal)
goods (location: New York)
transport (status: normal)
ship (status: normal)
f TRANSPORT
state
agent HAL (status: normal)
goods (location: Rotterdam)
transport (status: normal)
ship (status: normal)

Fig. 1 An action and its state transformation

To distinguish the different types of parameters in a more clear way, we will place semi-colons between them when specifying the function, e.g.:

$$f^{\text{TRANSPORT}}(transport : \mathcal{S};\ HAL : \mathcal{A};\ ship : \mathcal{E};\ goods : \mathcal{N}).$$

Figure 1 gives an illustration of the our representation of a plan step. It show the application of the plan step 'transport' which is a an instance of a TRANSPORT plan operator. The effect of executing the plan step is that the location of the goods changes. If necessary, we could also model that the location of the ship and of the executing agent HAL changes too.

The result of a plan step may not always be known if, for instance, the action fails, the agent misbehaves or the equipment malfunctions. Therefore we allow that the plan operator function associated with a plan step maps the value of a variable to $\perp$ to denote that the effect of the plan step on this variable is unknown. We will use the health mode *ab* to indicate the most general abnormal health mode of a plan step, an agent or equipment. If one of the input parameters of f^o has the health mode *ab*, the outcome of the action execution is assumed to be unknown. That is,

$$f^o(s, a_1, \dots, a_i, e_1, \dots, e_j, n_1, \dots, n_k) = (\perp, \dots, \perp)$$

if $s = ab$, for some l: $a_l = ab$, or for some l: $e_l = ab$.

In [30,24], it was assumed that the result of a plan step s is unknown (all variables in its range will take the value $\perp$) if one of the variables in the domain $v \in dom_{\mathcal{V}}(s)$ has an unknown value. Here, we relax this restriction allowing the function f^o with $s \in inst(o)$ to map to known values if some values in its (environment) domain are unknown.

3.5 Plans

A plan is a tuple $P = \langle S, < \rangle$ where $S \subseteq \mathcal{S}$ is a subset of the plan steps that need to be executed and $<$ is a partial order defined on $S \times S$ where $s < s'$ indicates that the plan step s must finish before the plan step s' may start. Note that each plan step $s \in S$ occurs exactly once in the plan P. We will denote the *transitive reduction* of $<$ by $\ll$, i.e., $\ll$ is the smallest sub-relation of $<$ such that the transitive closure $\ll^+$ of $\ll$ equals $<$.

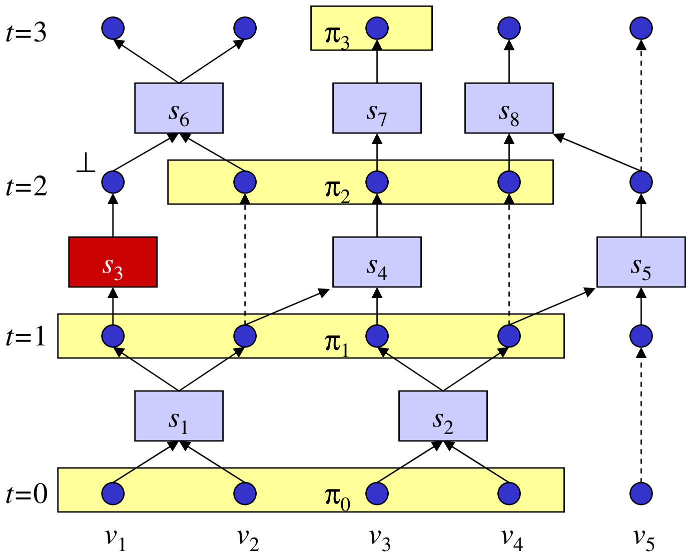

Fig. 2 Plan execution with one abnormal action

We assume that if in a plan P two plan steps s and s' are independent, in principle they may be executed concurrently. This means that the precedence relation $<$ at least should capture all resource dependencies that would prohibit concurrent execution of plan steps. Specifically, this implies that the precedence relation $<$

1. should prohibit simultaneous changes of the same variable v by different plan steps and
2. should prohibit changing a variable v by some plan step s while v may also be used as the input of another plan step s'.

Therefore, we assume $<$ to satisfy the following *concurrency requirement*[4]:

$$\text{If } ran_{\mathcal{V}}(s) \cap (dom_{\mathcal{V}}(s') \cup ran_{\mathcal{V}}(s)) \neq \varnothing \text{ then } s < s' \text{ or } s' < s.$$

Figure 2 gives an illustration of a plan with one abnormally executed plan step. Since a plan step is applied only once in a plan, for clarity reasons, we replace the function describing the behavior of the corresponding plan operator by the name of the plan step and we use the color of the plan step to denote its health mode. The arrows relate a plan step s_i to the variables it uses as inputs and the variables it modifies as its outputs. In this plan, the dependency relation is specified as $s_1 \ll s_3$, $s_1 \ll s_4$, $s_2 \ll s_4$, $s_2 \ll s_5$, $s_4 \ll s_7$, $s_5 \ll s_8$ and $s_4 \ll s_6$. The last dependency has to be included because s_6 changes the value of v_2 needed by s_4. The step s_4 shows that not every variable occurring in the domain of a plan step need to be affected by the plan step. Note that in this example the effects of plan steps s_6 (variables v_1 and v_2) and the effect of s_8 (variable v_4) are unknown because in the example we assume that all input variables of a plan step must be known in order to to get a known effect. In general, this may depend on the plan step.

[4] The simplifying assumption that will be made in the next paragraph makes it possible to simplify the concurrency requirement. In [25], the simplified concurrency requirement is called *Determinism*.

3.6 The relation with STRIPS

The most well known representation of plans is the one introduced with STRIPS [13]. STRIPS uses a set of ground instances of atomic propositions to describe the state of the world. In the above proposed formalism, we can represent this set of ground instances by introducing a variable for every ground instance of an atomic proposition and by assigning it the variable true if it is in the set describing the current state of the world and false otherwise.[5]

Plan steps in STRIPS change the truth values of some ground instances of atomic propositions. The functions that are associated with plan steps in the above proposed formalism do exactly the same thing. Given the values of some variables, which may represent the truth values of ground instances of atomic propositions, the values of a possibly different set of variables set; for instance to true or false.

Hence, a plan in the STRIPS notation can be can be transformed to the here proposed formalism.

3.7 Plan execution

For simplicity, we will assume that every plan step in a plan P takes one time unit to execute. We are allowed to observe the execution of a plan P at discrete times $t = 0, 1, 2, \ldots, k$ where k is the depth of the plan, i.e., the longest $<$-chain of plan steps occurring in P. Let $depth_P(s)$ be the depth of plan step s in plan $P = \langle A, < \rangle$. Here, $depth_P(s) = 0$ if $\{s' \mid s' \ll s\} = \varnothing$ and $depth_P(s) = 1 + max\{depth_P(s') \mid s' \ll s\}$, otherwise. If the context is clear, we often will omit the subscript P. We assume that the plan starts to be executed at time $t = 0$ and that concurrency is fully exploited, i.e., if $depth_P(s) = k$, then execution of s has been completed at time $t = k + 1$. Thus, all plan steps s with $depth_P(s) = 0$ are completed at time $t = 1$ and every plan step s with $depth_P(s) = k$ will be started at time k and will be completed at time $k + 1$. It is not difficult to see that, thanks to the above specified concurrency requirement, concurrent execution of plan steps having the same depth leads to a well-defined result.

To model plan execution, we need the notion of a *timed state*. A timed state is a tuple (π, t) where π is a state and $t \geq 0$ a time point. Now, given some timed state (π, t), let us consider the timed state (π', t') that results by executing plan P on (π, t). To define this plan execution relation in a precise way, we first need to define the set of plan steps that can be executed at some time t. But this is easy, since this is the set P_t of all plan steps s with $depth_P(s) = t$.

Now we can predict the timed state $(\pi', t + 1)$ using the timed state (π, t) and the set P_t of plan steps to be executed at time t as follows:

- For every plan step $s \in P_t$, the function f^o with $s \in inst(o)$ determines the values of the variables $v \in ran_{\mathcal{V}}(s)$ bound to the range parameters of f^o.
- All remaining variables v not in the range of a plan step $s \in P_t$ do not change their value between time points t and $t + 1$.

Note that the concurrency requirement guarantees that the values of the variables at time point $t + 1$ are well defined.

The following definition formalizes the result of executing the plan steps in P_t:

Definition 2 We say that $(\pi', t + 1)$ is (directly) generated by execution of P from (π, t), abbreviated by $(\pi, t) \rightarrow_P (\pi', t + 1)$, if the following conditions hold:

[5] Note that the proposed transformation need not always be the best possible. It might for instance be better to introduce a location-variable for every object mentioned in the predicate: $location(?object, ?position)$.

1. $\pi'(v) = f^o(\pi \restriction dom_{\mathcal{V}}(s))(v)$ for each $s \in P_t$ with $s \in inst(o)$ and for each $v \in ran_{\mathcal{V}}(s)$.
2. $\pi'(v) = \pi(v)$ for each $v \notin \bigcup_{s \in P_t} ran_{\mathcal{V}}(s)$, that is, the value of any variable not occurring in the range of a plan step in P_t should remain unchanged.

We extend this direct derivability relation to a general derivability relation in a straightforward way:

Definition 3 For arbitrary values of $t \leq t'$, (π', t') is said to be (directly or indirectly) generated by execution of P from (π, t), denoted by $(\pi, t) \rightarrow_P^* (\pi', t')$, iff the following conditions hold:

1. if $t = t'$ then $\pi' = \pi$;
2. if $t' = t + 1$ then $(\pi, t) \rightarrow_P (\pi', t')$;
3. if $t' > t + 1$ then there exists some state $(\pi'', t' - 1)$ such that $(\pi, t) \rightarrow_P^* (\pi'', t' - 1)$ and $(\pi'', t' - 1) \rightarrow_P (\pi', t')$.

Note that the semantics of a plan execution corresponds to the Hoare semantics of programming languages [14].

3.8 Dynamic changes of health modes

In [30,24], Witteveen et al. describe how plan execution can be diagnosed by viewing a plan step s of a plan as a component of a system having a *normal* or an *abnormal* behavior, and by viewing the input and output variables of s as in- and outputs of a component. This view enabled them to apply classical MBD to plan execution, and to characterize a diagnosis simply as a subset $Q \subseteq S$ of abnormally functioning components, i.e., abnormally executed plan steps. Here, we extend the set of possible diagnoses considered by these authors. First of all, we allow for more health modes than just *nor* and *ab*. Secondly, besides health modes for plan steps, we also introduce health modes for agents, equipment and some of the objects in the environment. Moreover, we consider the health modes of agents, equipment and objects in the environment as dynamic variables that may change over time. For instance, consider the health mode of an airplane's engine. It may function *normally* at the start of plan, may become *overheated* during take-off and may *brake-down completely* during the flight. Of course, not every transition between these health modes is valid. For example, an airplane with a broken engine cannot become an airplane with only a flat tyre without first repairing the engine. Therefore, considering health modes as dynamic variables, we have to specify *when* such a change occurs and also *which transitions* from one value to another value are possible. We will specify a health mode change by an *event* and possible heath modes changes by a *transition function*.

An event is a triple (v_j, d, t) specifying that the variable v_j takes the value $d \in D_{v_j}$ at the time point t. Hence, a variable v_j, such as a plan step, an agent or a piece of equipment, is qualified by the health mode d at time point t.

Possible transitions between health mode assignments for a variable v are specified by a transition function $tr_v \colon D_v \rightarrow 2^{D_v}$ that given a current value d of v specifies the set of allowable transitions of the health mode d to other health modes of v. Note that we have assumed that values of variables *not* describing health modes, such as the location of an airplane, may only change their values as the result of executing plan steps. Therefore, for these variables no events changing their values are allowed. By choosing for such variables v the empty transition function: $tr_v(d) = \varnothing$, for every $d \in D_v$, we can assure that no event

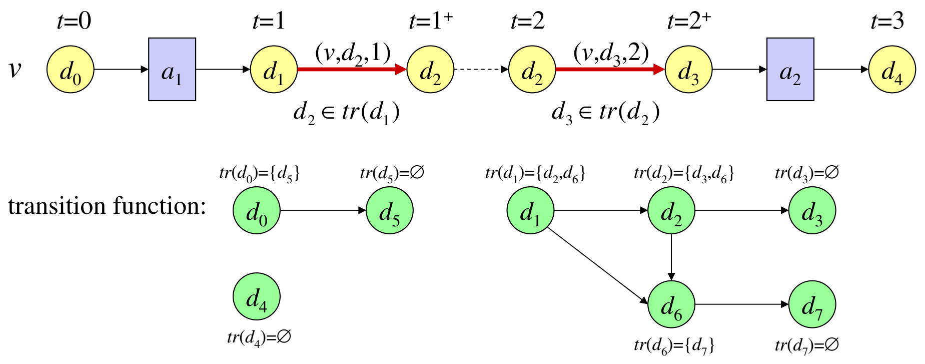

Fig. 3 A Discrete Event System of the variable v

changes the state of the variable v. Hence, we can simply assume that a transition function tr_v is specified for every variable v.

Remark 4 Note that we might view each variable v_j as representing a DES [5]. In this DES, a triple (v_j, d, t) describes an unknown *event* that changes the state of the variable v_j, and the transition function $tr_j\colon D_j \to 2^{D_j}$ describes the transition allowed by DES. Figure 3 gives an illustration. The goal of a diagnosis in such systems is to identify these unknown events (v_j, d, t) that have caused the state changes. Also note that plan steps enforce state changes independently of the transition function tr_j.

3.9 Qualifications

In a previous section we have specified the plan derivability relation $\rightarrow_P^*$ that enabled us to predict a timed state from an earlier timed state not taking into account unforeseen changes in the values of health modes of variables. Since we have used events to specify such changes, we will now incorporate the specification of such events in the definition of the plan derivability relation.

First of all, we define a plan *qualification*, denoted by κ, as a set of events changing the states of some variables v at specific time points t and thereby causing specific state changes in the plan. Using these qualifications, we (re)define the derivability relation:

Definition 4 Let $(\pi'', t) \rightarrow_P (\pi', t+1)$ be the direct derivability relation given a plan P, and let κ be a qualification.

We say that $(\pi', t+1)$ is (directly) generated by execution of P from (π, t) given the qualification κ, abbreviated by $(\pi, t) \rightarrow_{\kappa;P} (\pi', t+1)$, iff there exists a state π'' such that:

1. for each $v_j \in \mathcal{V}$: $\pi''(v_j) = d$ if $(v_j, d, t) \in \kappa$, and $\pi''(v_j) = \pi(v_j)$ otherwise;
2. $(\pi'', t) \rightarrow_P (\pi', t+1)$.

Note that we use an implicit transition from time point t to $t + \varepsilon$ with ε approximating 0 in the definition. During this transition, the variable v_j takes the value d.

Again we extend this direct derivability relation to a general derivability relation in a straightforward way to $(\pi, t) \rightarrow_{\kappa;P}^* (\pi', t')$.

In the above defined derivability relation given a qualification κ, we did not take into account whether the state changes specified by κ are unambiguous and are allowed according to the transition functions $tr_v\colon D_v \to 2^{D_v}$. If the state changes are allowed, we say that

κ induces a *sound* derivation. Of course, soundness of such a derivation given qualification κ not only depends on the set of events described by κ but also on the observed state π of the world from which the derivation starts and the plan P that is executed.

Definition 5 Let κ be a qualification and let tr_v: $D_v \to 2^{D_v}$ be the transition function of the variable v. We say that a qualification κ induces a *sound* derivation $(\pi, t) \to^*_{\kappa;P} (\pi', t')$ iff:

– (no ambiguity) for no pair of events (v_j, d, t_1), $(v_j, d', t_2) \in \kappa$, we have that $t_1 = t_2$, and
– (allowed transition) for every $t \leq t'' < t'$ and for event $(v, d, t'') \in \kappa$, if $(\pi, t) \to^*_{\kappa;P} (\pi'', t'') \to^*_{\kappa;P} (\pi', t')$, then $d \in tr_v(\pi''(v))$.

3.10 Default assumptions and plan execution

To predict the effect of executing a plan, we usually start with a partial observation π of the state of the world at some time point t. This raises a problem with respect to the health modes of plan steps, agents, equipment and possibly some environment variables. Often, these variables v do not belong to the set of observed variables and therefore will be undefined in π. Nevertheless, knowledge of the values of these variables is essential for predicting the specific effects of executing the plan using the derivation relation. Therefore, we assume that each plan step, agent and equipment variable has a default value specified by a default function δ. Often the default value will be the value *nor*. Note that δ maps a variable without a default value to $\perp$. The default values $\delta(v)$ can be used to extend a partial observation π to a partial state denoted by $\delta(\pi)$ where default values are assigned to unobserved variables.

Definition 6 Let π be a partial state and let δ: $\mathcal{V} \to \bigcup_{v \in \mathcal{V}} D_v$ be a default function specifying the default values of the health mode variables.

The partial state extended with default values, denoted by $\delta(\pi)$, is defined as:

$$\forall v \in \mathcal{V}\ [\delta(\pi)(v) = \delta(v) \text{ if } \pi(v) = \perp;\ \delta(\pi)(v) = \pi(v) \text{ otherwise}].$$

4 Plan diagnosis

As we stated in the introduction, we will distinguish two forms of plan diagnosis: primary diagnosis and secondary diagnosis. By making (partial) observations at different time points of the ongoing plan execution we may establish that there are discrepancies between the expected and the observed plan execution. Identifying these discrepancies is called *fault detection*. The identified discrepancies indicate that the results of executing one or more plan steps differs from the way they were planned. Identifying these plan steps and, if possible, what went wrong in the plan steps' execution will be called *primary plan diagnosis*.

Plan steps may fail because external factors such as changes in the environmental conditions (the weather), failing equipment or incorrect beliefs of agents. These external factors are underlying causes which are important for predicting how the remainder of a plan will be executed. The *secondary plan diagnosis* aims at establishing these underlying causes.

We denote an instance of a primary or secondary diagnosis problem by the tuple:

$$\langle P, \delta, tr, obs(t), obs(t') \rangle$$

where $P = (S, <)$ is a plan description, δ is a function describing the default values of variables, tr is a set of transition functions tr_v: $D_v \to 2^{D_v}$ for every $v \in \mathcal{V}$, and $obs(t) = (\pi, t)$ and $obs(t') = (\pi', t')$ are observations of the partial states π and π' at time points t and t', respectively, where $0 \leq t < t' \leq depth(P)$.

Given such an instance, we would like to identify a suitable qualification κ such that the predicted state π'_κ using the derivation $(\delta(\pi), t) \rightarrow^*_{\kappa;P} (\pi'_\kappa, t')$ corresponds to the observation (π', t').

We will now discuss suitable qualifications κ to characterize primary and secondary diagnoses.

4.1 Primary plan diagnosis

In [30,24], we describe how plan execution can be diagnosed by viewing plan steps of a plan as components of a system and by viewing the input and output variables of a plan step as in- and outputs of a component. In this approach, a qualification $Q \subset S$ is a subset of plan steps s qualified as abnormal. Here, such a set Q can be represented by a qualification $\kappa_Q = \{(s, ab, depth(s)) \mid s \in Q\}$. A qualification consisting of a set of events $(s, d, depth(s))$ with $s \in S$ will be called a *plan step qualification*. To illustrate such a plan step qualification, consider Fig. 2. Suppose plan step s_3 is abnormal and generates a result that is unpredictable ($\perp$). Given the qualification $\kappa = \{(s_3, ab, 1)\}$ and the partially observed state π_0 at time point $t = 0$, we predict the partial states π_i as indicated in Fig. 2, where $(\pi_0, t_0) \rightarrow^*_{\kappa;P} (\pi_i, t_i)$ for $i = 1, 2, 3$. Note that in the example we assume that all inputs of a plan step must have known values in order to have predictable outputs. This implies that the results of plan steps s_6 and s_8 cannot be predicted because the values of v_1 and of v_5 cannot be predicted at time $t = 2$. Therefore, π_3 contains only the value of v_3.

Now consider a plan-diagnosis instance $\langle P, \delta, tr, obs(t), obs(t')\rangle$. In order to obtain a suitable primary diagnosis, we would like to use the observations $obs(t) = (\pi, t)$ and $obs(t') = (\pi', t')$ to infer the health modes of the plan steps occurring in $P = (S, <)$. Assuming a normal execution of P using the function δ to specify default values for variables, we can (partially) predict the state of the world at a time point t' given the observation $obs(t)$: if all plan steps behave normally, we predict a partial state $\pi'_\varnothing$ at time t' such that $(\delta(\pi), t) \rightarrow^*_P (\pi'_\varnothing, t')$. Therefore, if this assumption holds, the values of the variables that occur in both the predicted state and the observed state at time t' should match, i.e., there should exist a complete state σ, such that $\pi' \sqsubseteq \sigma$ and $\pi'_\varnothing \sqsubseteq \sigma$. But that implies that $\pi' \approx \pi'_\varnothing$. If, however, this is not the case, the execution of some plan steps s must have gone wrong and we have to determine a plan step qualification κ such that the predicted state derived using κ agrees with π'. This is nothing else than a straight-forward extension of the diagnosis concept in MBD to plan diagnosis (cf. [23,7]):

Definition 7 Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be a diagnostic instance. Moreover, let the plan step qualification κ be a set of triples $(s, d, depth(s))$ with $s \in \mathcal{S}$ and $d \in D_s$, and let κ induce a *sound* derivation $(\delta(\pi), t) \rightarrow^*_{\kappa;P} (\pi'_\kappa, t')$ given the plan P and a transition function $tr_s \colon D_s \rightarrow 2^{D_s}$ for each plan step $s \in S$.

Then κ is said to be a *primary plan diagnosis* (plan step diagnosis) of a plan-diagnosis instance $\langle P, \delta, tr, obs(t), obs(t')\rangle$ iff $\pi' \approx \pi'_\kappa$.

So in a primary plan diagnosis κ, the observed partial state π' at time t' and the predicted state π'_κ at time t' assuming the plan step qualification κ agrees upon the values of all variables $V(\pi') \cap V(\pi'_\kappa)$ occurring in both states.

Example 1 Consider again Fig. 2 and suppose that we did not know that plan step s_3 was abnormal and that we observed $obs(0) = ((d_1, d_2, d_3, d_4), 0)$ and $obs(3) = ((d'_1, d'_3, d'_5), 3)$. Using the normal plan derivation relation starting with $obs(0)$ we will predict a state $\pi'_\varnothing$ at time $t = 3$ where $\pi'_\varnothing = (d''_1, d''_2, d''_3)$. If everything is ok, the values of the variables predicted as well as observed at time $t = 3$ should correspond, i.e., we should have $d'_j = d''_j$ for $j = 1, 3$. If, for example, only d'_1 would differ from d''_1, then we could qualify s_6 as abnormal, since then the predicted state at time $t = 3$ using $\kappa = \{(s_6, ab, 2)\}$ would be $\pi'_\kappa = (d''_3)$ and this partial state agrees with the observed state on the value of v_3.

Note that for all variables in $V(\pi') \cap V(\pi'_\kappa)$, the qualification κ provides an *explanation* for the observation π' made at time point t'. Hence, for these variables the qualification provides an *abductive diagnosis* [6]. For all observed variables in $V(\pi') - V(\pi'_\kappa)$, no value can be predicted given the qualification κ. Hence, by declaring them to be unpredictable, possible conflicts with respect to these variables if a normal execution of all plan steps is assumed, are resolved. This corresponds with the idea of a *consistency-based diagnosis* [23].

4.2 Secondary plan diagnosis

Plan steps may fail because of unforeseen (environmental) conditions such as being struck by lightning, malfunctioning equipment or incorrect beliefs of agents. Secondary plan diagnosis aims at identifying the underlying causes of plan step failures. Therefore, secondary plan diagnosis only considers qualifications that change the values of variables *not* representing plan steps, i.e., the set of variables in $\mathcal{V} - \mathcal{S}$.

A *secondary qualification* κ then is a qualification consisting of triples (v, d, t), where $v \in \mathcal{V} - \mathcal{S}$. Note that there may be more than one time point between the execution of the plan step s where the value of the variable v must have been changed by an event and the latest plan step $s' < s$ where v was actually used by s', that is $v \in ran_{\mathcal{V}}(s') \cap dom_{\mathcal{V}}(s')$. Since there is no way to determine at which time point between $dept(s') + 1$ and $depth(s)$ the event (v, d, t) occurred, we usually choose for t the depth $depth(s)$ of the first plan step s where the change manifests itself.

Definition 8 Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be a diagnostic instance. Moreover, let the plan step qualification κ be a set of triples (v, d, t) with $v \in \mathcal{V} - \mathcal{S}$ and $d \in D_v$, and let κ induce a *sound* derivation $(\delta(\pi), t) \rightarrow^*_{\kappa;P} (\pi'_\kappa, t')$ given the plan P and a transition function $tr_j \colon D_j \rightarrow 2^{D_j}$ for each variable $v_j \in \mathcal{V}$.

Then the qualification κ is said to be a *secondary plan diagnosis* of a plan-diagnosis instance $\langle P, \delta, tr, obs(t), obs(t')\rangle$ iff $\pi' \approx \pi'_\kappa$.

The secondary diagnosis can be divided into *agent*, *equipment* and *environment diagnosis* depending on whether the variable v in a triple $(v, d, t) \in \kappa$ belongs to $\mathcal{A}$, $\mathcal{E}$ or $\mathcal{N}$, respectively. Note that agent diagnosis is related to *social diagnosis* described by Kalech and Kaminka [16,17] if the agents' health modes are used to describe the agents' incorrect beliefs.

4.3 Predicting the future

Secondary diagnosis offers an important advantage over primary diagnosis. First of all, secondary diagnosis enables us to determine which future plan steps may also be affected by the malfunctioning agents and equipment, and by unforeseen state changes in the environment. If a piece of equipment breaks down, then all plan steps that require this piece of equipment will fail. The state change described by a failure event $(v, d, t) \in \kappa$ persists: $\pi(v) = d$ at all time points $t' \geq t$ until another event or a plan step changes v again.

Definition 9 Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be a diagnostic instance and let κ be a current secondary diagnosis of the plan executed so far. Moreover, let t' be the current time point.

Then the set of future plan steps that will directly be affected given the current diagnosis κ is:

$$\{s \in \mathcal{S} \mid v \in dom_{\mathcal{V}}(s), (v, d, t'') \in \kappa, d \neq nor, t \leq t'' < t' \leq depth(s)\}$$

Second, besides identifying the plan steps that will be affected, we can also determine the goals that can still be reached.

Definition 10 Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be a diagnostic instance with $obs(t') = (\pi', t')$, and let κ be a current secondary diagnosis of the plan executed so far. Moreover, let t' be the current time point and let $(\delta(\pi)', t') \rightarrow_{\kappa;P} (\pi'', depth(P))$.

Then the set of goals that can still be realized is given by:

$$\{g \in G \mid \pi'' \models g\}$$

4.4 Responsible agents

Besides knowing the underlying cause of plan execution failures, it is also important to know the agents responsible for the failures. To illustrate this, reconsidering the example in the introduction where the agent responsible for the failing luggage sorting action may be the maintenance agent or the airport authorities that reduced the maintenance budget.

We might introduce a responsibility function res mapping a failed plan step, a misbehaving agent and malfunctioning equipment to a responsible agent. Such a function cannot handle more sophisticated forms of responsibility. For instance, a pilot being responsible for a rough landing and the maintenance personnel being responsible for a tire blow-out during landing. In both cases the landing plan step is executed in an abnormal way but the responsible agent depends on type of abnormal execution. Therefore, we propose a mapping based on a variable and its health mode.

Definition 11 Let κ be any diagnosis of a plan execution and let

$$res\colon \mathcal{V} \times \bigcup_{v \in \mathcal{V}} D_v \rightarrow \mathcal{A}$$

be function assigning responsibility to agents in $\mathcal{A}$.

Then for each event $(v, d, t) \in \kappa$, the responsible agent is determined by: $res(v, d)$.

This failure dependent responsibility may not suffice in all cases. Similar to primary and secondary diagnosis, we may introduce different levels of responsibility. An air traffic controller making an error can be primary responsible for the error. His or her supervisor may be secondary responsible for the error. The airport may be responsible at the third level while the aviation authorities may be responsible at the fourth level. Here, we leave such sophisticated models of responsibility for future research.

5 An example

This section will illustrate the diagnostic model presented in the previous section using a plan for transporting luggage of a passenger form one airport to another. We add to this plan the observation that luggage is dropped-off at the check-in and the observation at the destination indicating whether the luggage has arrived. Sometimes there is a discrepancy between our expectation about our luggage and that what we observe at the destination. We will use such a discrepancy to illustrate the diagnostic model presented in this paper.

5.1 The plan

Figure 4 depicts such a plan at an abstract level. Note that only the states of the variables 'luggage' and 'label' are changed by the plan steps. The state of the other variables are not changed by the plan steps, they are only used as inputs. Based on the graphical representation of a plan introduced in the previous section, all the variables in the columns to the right of the variables 'luggage' and 'label' should actually be placed alongside the variables 'luggage' and 'label'. The variables of which the states are not changed by the execution plan steps are placed near the plan steps that use them as inputs because of the limited width of the page. Note that we have ordered the variables to the right of the variables 'luggage' and 'label'

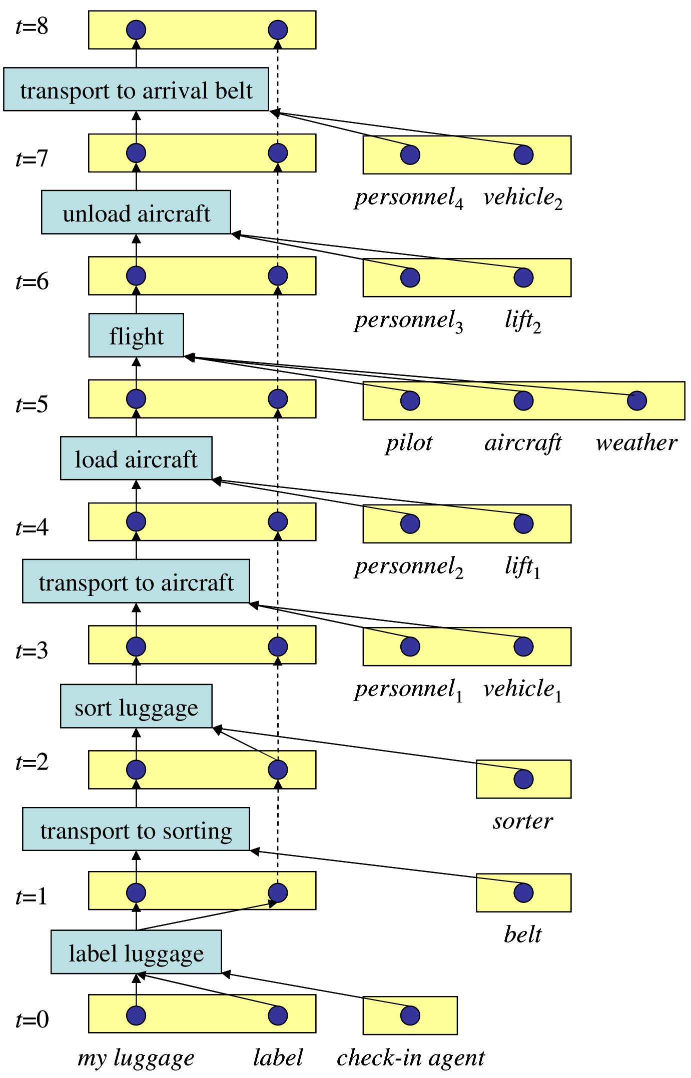

Fig. 4 A plan for transporting a passenger's luggages

in columns: first a column with agents, then a column with equipment and finally a column with other environment variables.

The plan in Fig. 4 only shows the transport of one piece of luggage. An airport handles of course many pieces of luggage, some for the same flight and some for other flights. For each piece of luggage there is a similar plan which may require the same plan steps, the same agents and the same equipment during its execution. Together these individual plans form the total plan of luggage handling on an airport.

5.2 Primary diagnosis

Suppose that at our destination we observe the absence of our luggage. Using the plan depicted in Fig. 4, we may determine as our *primary diagnosis* that one of the eight plan steps must have been executed *abnormally*:

$$\begin{aligned}
\kappa_1 &= \{(\text{`label luggage'}, ab, 0)\},\\
\kappa_2 &= \{(\text{`transport to sorting'}, ab, 1)\},\\
\kappa_3 &= \{(\text{`sort luggage'}, ab, 2)\},\\
\kappa_4 &= \{(\text{`transport to aicraft'}, ab, 3)\},\\
\kappa_5 &= \{(\text{`load airplane'}, ab, 4)\},\\
\kappa_6 &= \{(\text{`flight'}, ab, 5)\},\\
\kappa_7 &= \{(\text{`unload aircraft'}, ab, 6)\},\\
\kappa_8 &= \{(\text{`transport to arrival belt'}, ab, 7)\}
\end{aligned}$$

Note that these are all single fault diagnoses; i.e., each diagnosis consist of a single failure event. Intuitively, we prefer these diagnoses over diagnoses with multiple faults. In the next section, we will formalize the intuition.

Having observed the absence of our luggage, we may go to the help desk where, using a communication network, the service person finds out that your luggage has arrived in Amman (AMM) instead of Amsterdam (AMS). Based on fault models of the different plan steps, we may determine better primary diagnoses. For instance, the plan step 'sort luggage' may have routed the luggage to the wrong loading platform.

$$\kappa = \{(\text{`sort luggage'}, \text{`routing error: AMS} \rightarrow \text{AMM'}, 2)\}$$

Usually finding back someone's lost luggage will take several hours. Even before information about the luggage's location come available, we can already determine a more specific set of primary diagnoses. The transport plan of our luggage is a sub-plan of a much larger plan. In this larger plan, other passengers have also transported their luggage using the same flight. If the luggage of some of these passengers have correctly arrived in Amsterdam, we know that the plan step 'flight' is executed normally. (Of course, we already knew this because we arrived ourselves at the destination.) Moreover, if the plan steps 'transport to airplane', 'load airplane' 'unload aircraft' and 'transport to arrival belt' are the same plan steps for all the luggage on a flight, then these plan steps have not failed either. Therefore, using the larger plan also enables us to determine a more specific set of primary diagnoses:

$$\begin{aligned}
\kappa_1 &= \{(\text{`labeling luggage'}, ab, 0)\},\\
\kappa_2 &= \{(\text{`transport to sorting'}, ab, 1)\},\\
\kappa_3 &= \{(\text{`sort luggage'}, ab, 2)\}
\end{aligned}$$

5.3 Secondary diagnosis

Beside identifying the plan steps that may have failed, we may also consider malfunctioning equipment that may have caused the plan steps to fail. For instance, a malfunctioning sensor on the ‘sorter’ may be responsible for routing some of the luggage to the wrong platform. The *equipment diagnosis*

$$\kappa = \{(\text{‘sorter’}, \text{‘sensor error: } S \rightarrow M\text{’}, 2)\}$$

explains why some pieces of luggage ended up in Amman (AMM) instead of Amsterdam (AMS). It also explain why some pieces of luggage ended up in Laiagam (LGM) instead of Malargue (LGS). Moreover it enables us to *predict* that some pieces of luggage will ended up in Hvammstangi (HVM) instead of Municipal (HVS) .

Misbehaving agents may also result in a plan execution failure. The check-in agent may make errors with the label of the luggage and the personnel transporting the luggage to the airplane may forget a pieces of luggage. *Agent diagnosis* can be used to identify failing agents.

$$\kappa_1 = \{(\text{‘check-in agent’}, ab, 0)\},$$
$$\kappa_2 = \{(\text{‘personnel}_1\text{’}, ab, 3)\}$$

A damaged label may also explain why our luggage ended up in Amman instead of Amsterdam. *Environment diagnosis* can be used to identify this explanation for our lost luggage.

$$\kappa = \{(\text{‘label’}, \text{‘damaged’}, 2)\}$$

A change in the environment object ‘weather’ may also explain why our luggage did not reach its destination. Unforseen bad weather conditions may force an airplane to deviate to another airport. This is, however, an adaptation of the plan instead of an incorrect execution of the plan. The plan diagnosis approach described in the paper cannot be used to identify changes in the plan. It must know the plan that is actually executed. Therefore, this type of cause cannot be handled by the diagnosis of plan execution we describe here.

5.4 Responsible agents

Usually the agent(s) executing a plan step are responsible for the proper execution of the plan step. Therefore, for every failing plan step, we may assign the responsibility to the executing agents mentioned in Fig. 4. In much the same way we may assign responsibility for failing equipment to either the agent operating it or the agent maintaining it, depending on the type of fault that occurred. For instance, we can specify that ‘personnel$_1$’ is responsible for a ‘crash’ with ‘vehicle$_1$’ while ‘maintenance personnel’ is responsible for an ‘engine failure’ of ‘vehicle$_1$’.

In our application domain we are often more interested in who is financially responsible. This gives us a differen type of responsibility assignment. Airlines are responsible for plan executions of their airplanes, ground handling companies are responsible loading the airplanes, air traffic control is responsible for runways, taxi-ways, gate availabilities, and so on and so forth. These types of responsibilities can also be modeled using the responsibility function res.

6 Preference criteria on diagnoses

As we saw in the example presented in the previous section, both primary and secondary diagnosis need not provide a unique explanation of observed anomalies. When more than one diagnosis can be determined, the diagnoses need not all provide the same quality of explanation. Intuitively, a primary diagnosis in which one plan step is failing gives a better explanation than a primary diagnosis in which all plan steps are failing. This intuition has in fact been used in the previous section. In this section we formalize this intuition by defining preference criteria on diagnoses. Moreover, we analyze time complexity of identifying the preferred diagnoses.

6.1 The preference criteria

The most common preference criteria on diagnoses are (i) based on the subset relation with respect to the sets of health mode variables indicating failures (i.e., variables $v \in \mathcal{S} \cup \mathcal{A} \cup \mathcal{E}$ for which there are events $(v, d, t) \in \kappa$ such that $d \neq \delta(v)$), and (ii) the number of these health mode variables. The intuition behind the first preference criterium is that if failures occur independent of each other, and if the probabilities that a failure occurs is less than 0.5, a diagnosis that has a subset of the failures of another diagnosis has a higher probability of being correct. A diagnosis in which a subset minimal failures occurs is called *minimal diagnosis*:

Definition 12 Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be a diagnostic instance.
A diagnosis κ is a *minimal diagnosis* iff for no diagnosis κ':

$$\{v \in \mathcal{S} \cup \mathcal{A} \cup \mathcal{E} \mid (v, d, t) \in \kappa', d \neq \delta(v)\} \subset \{v \in \mathcal{S} \cup \mathcal{A} \cup \mathcal{E} \mid (v, d, t) \in \kappa, d \neq \delta(v)\}$$

The intuition behind the second preference criterium is based on a stronger requirement, namely that the failure probabilities are very small and that they are more or less of the same magnitude. In that case, the most probable diagnoses are those having a numerical minimum number of failures.

Definition 13 Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be a diagnostic instance.
A diagnosis κ is a *minimum diagnosis* iff for no diagnosis κ':

$$|\{v \in \mathcal{S} \cup \mathcal{A} \cup \mathcal{E} \mid (v, d, t) \in \kappa', d \neq \delta(v)\}| < |\{v \in \mathcal{S} \cup \mathcal{A} \cup \mathcal{E} \mid (v, d, t) \in \kappa, d \neq \delta(v)\}|$$

A third preference criterium on diagnoses is based on the explanative power for diagnoses [25]. Given an instance of a plan-diagnostic problem $\langle P, \delta, tr, obs(t), obs(t')\rangle$, a diagnosis κ may explain more observations at time point t' than a diagnosis κ'. If the probability that the observed value of a variable is correctly predicted given a diagnosis is very small, then the diagnosis κ has a higher probability of being correct than the diagnosis κ'. We say in such cases that the diagnosis κ is *more informative* than the diagnosis κ' [25]. Given this information order on diagnosis, we can define the *maximal-informative* diagnoses, which also turn out to be *maximum-informative* diagnosis.

Definition 14 Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be a diagnostic instance, and let π'_κ be defined as: $(\delta(\pi), t) \rightarrow^*_{\kappa; P} (\pi'_\kappa, t')$.
A diagnosis κ is said to be a *maximum-informative* diagnosis iff for no diagnosis κ':

$$V(\pi'_\kappa) \subset V(\pi'_{\kappa'})$$

Since a maximum-informative diagnosis need not be a minimal diagnosis, the two preference criteria can be combined resulting in *minimal maximum-informative* (*mini-maxi*) diagnoses.

6.2 Complexity issues

In [25] Roos and Witteveen have shown that identifying a minimum diagnosis is an NP-hard problem while identifying a minimal maximum-informative (mini-maxi) diagnosis can be done in polynomial time. Identifying a *minimum* maximum-informative diagnosis, however, is again an NP-hard problem. In this section, we briefly consider some complexity issues with respect to finding diagnoses in our framework, relating them to the results obtained in [25].

6.2.1 Finding minimum diagnoses

Identifying a minimum diagnosis is also an NP-hard problem for the plan diagnosis model presented in this paper: By only considering primary diagnoses and by restricting the health modes in a primary diagnosis to *nor* and *ab*, the diagnostic problem becomes identical to the diagnostic problem addressed in [25].

Proposition 1 *Finding minimum primary diagnoses is NP-hard.*

Proof We transform an instance $\langle P, obs(t), obs(t')\rangle$ of a diagnostic problem as described in [25] to a diagnostic instance $\langle P, \delta, tr, obs(t), obs(t')\rangle$ described in this paper. Note that a (minimum) solution to the former diagnostic problem is (minimum) set of plan steps Q qualified as being *abnormal*.

Given the instance $\langle P, obs(t), obs(t')\rangle$ with a set of plan steps S and a set of variables $\mathcal{V}$:

– Extend $\mathcal{V}$ with the variables $\{v_s \mid s \in S\}$.
– Replace every function $f_s^{nor}(v_1, \ldots, v_n)$ with $s \in inst(o)$ by the function $f^o(v_s : \mathcal{S}; v_1, \ldots, v_n : \mathcal{N})$ where $f^o(nor : \mathcal{S}; d_1, \ldots, d_n : \mathcal{N}) = f_s^{nor}(d_1, \ldots, d_n)$ and $f^o(ab : \mathcal{S}; d_1, \ldots, d_n : \mathcal{N}) = f_s^{ab}(d_1, \ldots, d_n) = (\perp, \ldots, \perp)$.
– Let δ be specified as follows: $\delta(v_s) = nor$ for each $s \in S$, and $\delta(v) = \perp$ otherwise.
– $tr = \{tr_{v_s} : s \in S\}$, where $tr_{v_s}(nor) = \{ab\}$ and $tr_{v_s}(ab) = \varnothing$.

Let $\langle P, \delta, tr, obs(t), obs(t')\rangle$ be the resulting diagnostic instance in our current framework. It is easy to see that a minimum primary diagnosis κ obtained for this instance corresponds to a minimum diagnosis $Q = \{s \mid (v_s, ab, depth(v_s)) \in \kappa\}$ in the framework of [25]. Hence, we can obtain a minimum plan diagnosis Q by reducing the original diagnostic problem to the primary diagnostic problem described in this paper. Therefore finding primary diagnoses in our current framework is NP-hard, too. □

6.2.2 Finding mini-maxi diagnoses

A *mini-maxi diagnosis* can be determined in polynomial time for both primary and secondary diagnosis. We will show this by simply adapting the polynomial algorithm given in [25]. Let $(\delta(\pi), t) \rightarrow^*_{\varnothing; P} (\pi'_\varnothing, t')$ be a prediction of the state at time point t' assuming the absence of failures (i.e.: $\kappa = \varnothing$) and let π' be the observed state at time point t'. To determine a maximum-informative diagnosis, we first determine the disagreement set V^{dif} of all those variables whose values are defined in both the observed state π' and the predicted state $\pi'_\varnothing$ at time t' but differ:

$$V^{dif} = \{v \in \mathcal{V} \mid \pi'_\varnothing(v) \neq \pi'(v), \pi'_\varnothing(v) > \perp, \pi'(v) > \perp\}.$$

Next, for $i = 1$, we collect all plan step variables $s \in \mathcal{S}$ such that $s \in P_{t'-i}$ and $ran_{\mathcal{V}}(s) \cap V^{dif} \neq \varnothing$. Subsequently, we add the qualification $(s, ab, t' - i)$ to κ_{max} and remove all variables $ran_{\mathcal{V}}(s)$ form the disagreement set V^{dif} for all the collected plan steps s. For $i = 2, 3, \ldots,$ we iteratively select new plan step variables $s \in \mathcal{S}$ at times $t' - i$ repeating the same procedure. It can easily be proven that this qualification κ_{max} is a maximum-informative diagnosis.

In order to obtain a mini-maxi diagnosis instead of just a maximum-informative diagnosis, we have to refine this procedure slightly using the notion of a *scope* of a variable v at time point t. The function $scope(v, t)$ returns the smallest set of variables that become undefined at time point t' if $\kappa = \{(v, ab, t'')\}$. If a heath mode variable is set to the value ab by an event at time point t'', then every variable in the range of a plan step $s \in P_{\geq t''}$ with $v \in dom_{\mathcal{V}}(s)$ may become undefined. Moreover if a variable in the range of a plan step $s \in P_{t^*}$ becomes undefined, then so may the variables in the range of the plan steps $s' \in P_{>t^*}$ with $ran_{\mathcal{V}}(s) \cap dom_{\mathcal{V}}(s') \neq \varnothing$.

We can use the function $scope(v, t)$ to remove events from κ_{max} till it becomes a minimal diagnosis. The resulting diagnosis is mini-maxi diagnosis. Since the whole procedure can be executed in polynomial time, a mini-maxi diagnosis can be identified in polynomial time.

Example To illustrate the identification of a mini-maxi diagnosis, we use the scenario described in Sect. 5, in which we observe at $t = 8$ that our luggage did not reach its destination; i.e., $V^{dif} = \{$'*myluggage*'$\}$. There is a plan step determining the value of the variable '*my luggage*' at $t = 7$, namely 'transport to arrival belt'. Hence, we add ('transport to arrival belt', ab, 7) to κ_{max}. Since V^{dif} contains only one variable,

$$\kappa_{max} = \{(\text{'transport to arrival belt'}, ab, 7)\}$$

is a *maximum-informative* diagnosis. Note that κ_{max} is also a *primary* diagnosis.

The maximum-informative diagnosis κ_{max} is also *mini-maxi* diagnoses because no event can be removed from κ_{max} without guaranteeing that all variables is V^{dif} are covered. Now suppose that we also have to qualify the plan step 'unload aircraft' as abnormal; i.e.

$$\kappa' = \{(\text{'unload luggage'}, ab, 6), (\text{'transport to arrival elt'}, ab, 7)\}$$

because of some variable v also belonging to V^{dif}. If every variable has an unpredictable effect whenever one of its inputs is undefined, then κ' is maximum-informative diagnoses but not mini-maxi diagnoses. Instead,

$$\kappa'' = \{(\text{'unload luggage'}, ab, 6)\}$$

is a *mini-maxi* diagnosis since we can eliminate ('transport to arrival belt', ab, 7) form κ' using the function $scope(v, t)$.

The above described procedure gives us one mini-maxi diagnosis for the example of Sect. 5. We can use the function $scope(v, t)$ to identify other mini-maxi diagnosis. Any diagnosis κ such that $\bigcup_{(v,ab,t)\in\kappa} scope(v, t) = \bigcup_{(v,ab,t)\in\kappa_{max}} scope(v, t)$ is also mini-maxi diagnosis. For instance, the diagnoses:

$$\begin{aligned}
&\kappa_1 = \{(\text{'personel}_4\text{'}, ab, 7)\},\\
&\kappa_2 = \{(\text{'vehicle}_2\text{'}, ab, 7)\},\\
&\kappa_3 = \{(\text{'unload luggage'}, ab, 6)\},\\
&\kappa_4 = \{(\text{'personel}_3\text{'}, ab, 6)\},\\
&\kappa_5 = \{(\text{'lift}_2\text{'}, ab, 2)\},\\
&\text{and so on, and so forth.}
\end{aligned}$$

Note that the diagnoses κ_1, κ_2, κ_4 and κ_5, are *secondary* diagnoses because the variables mentioned in the diagnoses either belong to $\mathcal{A}$ or to $\mathcal{E}$. Also note that the transition function $tr_v(\cdot)$ always allows a transition to *ab* for a plan step variable. For other variables such as 'personnel$_4$', we have to verify whether this transition is allowed given the value of the variable just before the event ('personel$_4$', ab, 7).

6.2.3 Dealing with additional health modes

Identifying a diagnosis in which health modes other than *nor* and *ab* are used is an NP-hard problem, even for mini-maxi diagnoses. This does not come as a surprise since identifying these diagnoses is a form of abduction, which is a well-known NP-hard problem. We will show that identifying an arbitrary diagnosis in which health mode variables are qualified with health modes other than *ab* is an NP-hard problem.

Proposition 2 *Identifying an arbitrary primary diagnosis is NP-hard.*

Proof To prove that a primary diagnosis is NP-hard, we reduce the NP-hard INTEGER PARTITION problem (Given a set I of integers and an integer r, find a subset of I whose sum equals r) to a primary diagnosis problem as follows:

Given an instance (I, r) of INTEGER PARTITION, construct an instance of a primary diagnosis problem as follows:

- Introduce a plan step variable s_i and a plan operator o_i for every integer $i \in I$.
- Introduce a new environment variable v.
- For each integer $i \in I$ introduce a function f^{o_i}, defined as: $f^{o_i}(nor, d) = (d + i)$ and $f^{o_i}(failure, d) = (d)$. The plan step s_i binds the parameters of f^{o_i}: $dom_{\mathcal{V}}(s_i) = (s_i, v)$ and $ran_{\mathcal{V}}(s_i) = (v)$.
- Let $tr_{s_j}(nor) = \{failure, ab\}$, $tr_{s_j}(failure) = \{ab\}$ and $tr_{s_j}(ab) = \varnothing$.
- Let $P = (S, <)$ be the plan, where $s_i < s_j$ iff $i < j$.
- Let $obs(0) = (\pi, 0)$ be an observation such that for every plan step $s \in S$: $\pi(s) = nor$ and for the variable $v \in \mathcal{N}$: $\pi(v) = 0$.
- Let the $obs(|I|) = (\pi', |I|)$ be an observation such that $\pi'(v) = r$.

It is not difficult to show that there exists a diagnosis κ such that $\pi_\kappa \approx \pi'$ iff the INTEGER PARTITION problem instance has a solution, i.e., a subset of integers that add up to r. Hence, finding a primary diagnosis is NP-hard. □

Proposition 3 *Identifying an arbitrary secondary diagnosis is NP-hard.*

Proof To prove that finding a secondary diagnosis is NP-hard, we reduce the 3-SAT problem to the secondary plan diagnosis problem. Consider a 3-SAT instance with variables $U = \{u_1, \ldots, u_n\}$ and clauses $C = \{c_1, \ldots, c_m\}$. Let $Lit(u_i) = \{u_i, \neg u_i\}$.

- For every clause $c \in C$, introduce a plan step variable s_c, a plan operator $o_c \in \mathcal{O}$ such that $s_c \in inst(o_c)$, and an environment variable n_c. Moreover, let $D_{n_c} = \{true, false\}$ be the domain of n_c.
- For every $u \in U$, introduce a variable v_u belonging to $\mathcal{X}$. Here, $\mathcal{X}$ is either $\mathcal{A}$, $\mathcal{E}$ or $\mathcal{N}$. Moreover, let $D_{v_u} = \{true, false\}$ be the domain of v_u.
- For each clause $c = (x_{i_1}, x_{i_2}, x_{i_3}) \in C$, where $x_{i_j} \in Lit(u_{i_j})$ for $j = 1, 2, 3$, the function f^{o_c} is defined as follows: $f^{o_c}(nor, d_1, d_2, d_3) = (true)$ iff a truth assignment I such that $I(u_{i_j}) = d_j$, for $j = 1, 2, 3$, satisfies c. The plan step s_c binds the parameters of f^{o_c}: $dom_{\mathcal{V}}(s_c) = (s_c, v_{u_1}, v_{u_2}, v_{u_3})$ and $ran_{\mathcal{V}}(s_c) = (n_c)$.

- Let $tr_{s_c}(nor) = \{ab\}$ and let $tr_{s_c}(ab) = \varnothing$.
- Let $tr_{n_c}(false) = tr_{n_c}(true) = \varnothing$.
- Let $tr_{v_u}(false) = \{true\}$ and let $tr_{v_u}(true) = \varnothing$.
- Let $P = (S, \varnothing)$ be the plan.
- Let the $obs(0) = (\pi, t)$ be an observation such that for every plan steps $s \in S$: $\pi(s) = nor$ and for every variable $v \in \mathcal{X}$: $\pi(v) = false$.
- Let the $obs(t) = (\pi', 1)$ be an observation such that for every variable n_c: $\pi'(n_c) = true$.

It is not difficult to show that there exists a secondary diagnosis κ such that $\pi_\kappa \approx \pi'$ iff the 3-SAT instance is satisfiable. Hence, agent, equipment and environment diagnosis are NP-hard. □

The above complexity results are of course disappointing. A mini-maxi diagnoses with only fault modes *nor* and *ab* can still be determined in polynomial time. However, identifying other fault modes than *ab* is NP-hard even for mini-maxi diagnoses.

In practice things are usually not that bad. First note that for any diagnosis κ there always exists a diagnosis $\kappa^* = \{(v, ab, t) \mid (v, d, t) \in \kappa, ab \in D_v\}$. We assume for the moment that agents also have health modes *nor* and *ab*. Since a mini-maxi diagnosis is a likely diagnosis, we may expect that κ^* is a mini-maxi diagnosis. We rather efficiently search for κ given κ^* if κ^* contains an event (v, ab, t) and there is a variable $v'' \in scope(v, t) \cap V^{dif}$ such that for no other event $(v', ab, t') \in \kappa^*$, $v'' \in scope(v', t')$. If this condition holds, we can test for all health modes $d \in D_v$ whether substituting (v, d, t) for (v, ab, t) in κ^* enables us to predict the observed values of the variables $(scope(v, t) - \bigcup_{(v',ab,t') \in \kappa^*, v \neq v'} scope(v', ab, t')) \cap V^{dif}$. If we can predict the values of these variables, then $\kappa^*[^{(v,d,t)}/_{(v,ab,t)}]$ is a diagnosis and we can continue searching for refinements of $\kappa^*[^{(v,d,t)}/_{(v,ab,t)}]$. So, we can repeat this procedure using the diagnosis κ^* in which (v, ab, t) is replaced by (v, d, t) using the disagreement set V^{dif} from which we have removed the variables $scope(v, t) - \bigcup_{(v',ab,t') \in \kappa^*, v \neq v'} scope(v', ab, t')$.

7 Conclusion

This paper describes a generalization of the model for plan diagnosis as presented in [30,24]. New in the current approach is (i) the introduction of the concepts of a primary and secondary diagnosis, (ii) the introduction of variables representing plan steps, agents and equipment, and (iii) the use of discrete event systems to describe the unknown dynamic behavior of variables such as equipment over time.

The primary diagnosis identifies failed plan steps and possibly in which way they failed, while the secondary diagnosis addresses the causes for plan step failures by identifying misbehaving agents and malfunctioning equipment. The latter is an improvement over the plan diagnosis presented in [30,24], where only dependencies between plan step failures could be described using causal rules. An additional feature of the proposed approach is the assignment of responsibilities for failures to specific agents. This responsibility assignment makes it possible, for instance, to divide cost of plan repair or to choose between plan repairs based on the impact on the agents involved.

Based on the results reported in [26], we have shown that finding minimum diagnoses is in general NP-hard. A minimal maximum-informative (mini-maxi) diagnosis can, however, still be determined in polynomial time. Unfortunately, however, identifying a mini-maxi diagnosis containing health modes other than *nor* and *ab* is again NP-hard. Finally, we pointed out conditions under which we may still identify a mini-maxi diagnosis containing health modes other than *nor* and *ab* efficiently.

Some of limitations of our model for plan diagnosis concern

– the identification of differences between the original plan and the executed plan; We assume that the plan is executed as specified. This assumption enables the identification of incorrect plan step executions, malfunctioning agents, broken equipment and unforeseen environment changes using partial observations of world at different time points.
– the identification of temporal constraint violations; To keep the discussion simple, in our model we assume that each plan step requires a unit time. The model also applies when plan steps have different durations. However, violations temporal constraints on the execution of a plan cannot be diagnosed using the current model.
– the modeling of plan steps as atomic entities. Often is suffices to model plan steps as atoms. However, it may sometimes be better to model an plan step as a process [20].

In future work we intend to extend our model cope with some of these limitations. First, we intend to incorporate diagnosis of *plan structure violations* into the model. Such plan structure violations are e.g., skipping a plan step, duplicating a plan step or swapping the order of two plan steps in the plan execution. Second, we wish to extend the model with diagnosis of temporal constraint violations. Here, we would like to refine our current plan description also with information about the time in which a certain plan step has to be executed.

Acknowledgements We thank the reviewers for their comments, which helped us to improve the paper. This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs (the Netherlands). Project DIT5780: Distributed Model Based Diagnosis and Repair.

References

1. Baroni, P., Lamperti, G., Pogliano, P., & Zanella, M. (1999). Diagnosis of large active systems. *Artificial Intelligence, 110*, 135–183.
2. Birnbaum, L., Collins, G., Freed, M., & Krulwich, B. (1990). Model-based diagnosis of planning failures. In *AAAI 90*, pp. 318–323.
3. Blum, A. L., & Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence, 90*, 281–300.
4. Carver, N., & Lesser, V. R. (2003). Domain monotonicity and the performance of local solutions strategies for cdps-based distributed sensor interpretation and distributed diagnosis. *Autonomous Agents and Multi-Agent Systems, 6*(1), 35–76.
5. Cassandras, C. G., & Lafortune, S. (1999). *Introduction to discrete event systems*. Kluwer Academic Publishers.
6. Console, L., & Torasso, P. (1990). Hypothetical reasoning in causal models. *International Journal of Intelligence Systems, 5*, 83–124.
7. Console, L., & Torasso, P. (1991). A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence, 7*, 133–141.
8. de Jonge, F., & Roos, N. (2004). Plan-execution health repair in a multi-agent system. In *PlanSIG 2004*.
9. de Jonge, F., Roos, N., & van den Herik, H. J. (2005). Keeping plan execution healthy. In *Multi-Agent Systems and Applications IV: CEEMAS 2005, LNCS 3690*, pp. 377–387.
10. de Kleer, J., Mackworth, A. K., & Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence, 56*, 197–222.
11. de Kleer, J., & Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence, 32*, 97–130.

12. McDermott, D., et al. (1998). The pddl planning domain definition language. In *The AIPS-98 Planning Competition Committee*.
13. Fikes, R. E., & Nilsson, N. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence, 5*, 189–208.
14. Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM, 12*(10), 576–585.
15. Horling, B., Benyo, B., & Lesser, V. (2001). Using self-diagnosis to adapt organizational structures. In *Proceedings of the 5th International Conference on Autonomous Agents* (pp. 529–536). ACM Press.
16. Kalech, M., & Kaminka, G. A. (2003). On the design of social diagnosis algorithms for multi-agent teams. In *IJCAI-03*, pp. 370–375.
17. Kalech, M., & Kaminka, G. A. (2005). Diagnosing a team of agents: Scaling-up. In *AAMAS 2005*, pp. 249–255.
18. Kalech, M., & Kaminka, G. A. (2007). On the design of coordination diagnosis algorithms for theams of situated agents. *Artificial Intelligence, 171*, 491–513.
19. Micalizio, R., & Torasso, P. (2007). On-line monitoring of plan execution: A distributed approach. *Knowledge-Based Systems, 20*, 134–142.
20. Micalizio, R., & Torasso, P. (2007). Team cooperation for plan recovery in multi-agent systems. In *Multiagent System Technologies, LNCS 4687*, pp. 170–181.
21. Pencolé, Y., & 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.
22. Poole, D. (1988). Representing knowledge for logic-based diagnosis. In *International Conference on Fifth Generation Computer Systems*, pp. 1282–1290.
23. Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence, 32*, 57–95.
24. Roos, N., & Witteveen, C. (2005). Diagnosis of plans and agents. In *Multi-Agent Systems and Applications IV: CEEMAS 2005, LNCS 3690*, pp. 357–366.
25. Roos, N., & Witteveen, C. (2006). Models and methods for plan diagnosis. In *Formal Approaches to Multi-Agent Systems (FAMAS'06).*
26. Roos, N., & Witteveen, C. (2008). Models and methods for plan diagnosis. *Journal of Autonomous Agents and Multi-Agent Systems*. doi:10.1007/s10458-007-9017-6.
27. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1995). Diagnosibility of discrete event systems. *IEEE Transactions on Automatic Control, 40*, 1555–1575.
28. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1996). Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology, 4*, 105–124.
29. Tonino, H., Bos, A., de Weerdt, M., & Witteveen, C. (2002). Plan coordination by revision in collective agent based systems. *Artificial Intelligence, 142*, 121–145.
30. Witteveen, C., Roos, N., van der Krogt, R., & de Weerdt, M. (2005). Diagnosis of single and multi-agent plans. In *AAMAS 2005*, pp. 805–812.