Temporal diagnosis of multi-agent plan execution without an explicit representation of time[1]

Femke de Jonge Nico Roos Huib Aldewereld

Dept of Computer Science, Universiteit Maastricht
P.O.Box 616, NL-6200 MD Maastricht
{f.dejonge,roos,h.aldewereld}@micc.unimaas.nl

Abstract

The most common reason for plan repair are the violation of a plan's temporal constraints. Air Traffic Control is an example of an area in which violations of the plan's temporal constraints is rather a rule than an exception. In such domains there is a need for identifying the underlying causes of the constraint violations in order to improve plan repairs and to anticipate future constraint violations. This paper presents a model for identifying the causes of the temporal constraint violations using coupled Discrete Event Systems. We show that for temporal diagnosis we can use a model without explicit temporal information.

1 Introduction

Violation of a plan's temporal constraints is one of the most common problems during plan execution. This holds especially in the domain of air traffic control. In this domain common causes of temporal constraint violations are problems in luggage handling, security issues, no-shows of passengers, unforseen changes in the weather conditions, and so on and so forth.

In order to repair plans, accurate information about the cause of the problem is important. It enables planners to come up with better plan repairs, thereby avoiding fire fighting tactics. This requires that we not only identify the *primary* cause, that is, failing plan step(s) causing constraint violations, but also the *secondary* cause, that is, failing equipment, unforseen changes in the environment and malfunctioning agents that are responsible for plan step failures.

In a recent paper the authors have proposed coupled Discrete Event System (DES) based on *timed automatons* to enable diagnosis of temporal constraint violations [6]. The use of timed automaton enables the generation of events if a DES is in a state for a specified period of time or if a DES is in a state at a specified time point. Although this approach makes it possible to accurately describe the normal and abnormal execution of a plan for the purpose of diagnosis, it has two important drawbacks. First, uncertainty with respect to the time at which events occur complicates the diagnosis problem. Second, existing methods for distributed diagnosis of coupled DESs cannot be used [1, 13].

In this paper we re-examine the model proposed in [6] and we propose to abstract from the specific time information. The abstraction will enable the application of currently available diagnosis techniques to diagnosis of temporal constraint violations during plan execution.

The remainder of the paper is organized as follows. Section 3 discusses how discrete event systems can be for modeling plan execution. Section 4 defines plan-execution diagnosis of temporal constraint violations and defines an explanation for constraint violations with respect to a diagnosis. Section 5 presents an example and Section 6 concludes the paper. First, to place our approach into perspective, we discuss some some related work.

[1]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.

2 Related work

In this section we briefly discuss some other approaches to plan diagnosis and subsequently some approaches to diagnosis using discrete event systems.

Plan diagnosis Birnbaum et al. [2] apply Model-Based Diagnosis (MBD) to *planning agents* relating health states of agents to *outcomes* of their planning activities. They do not take into account abnormalities that can be attributed to plan steps in a plan as a separate source of errors. In contrast to their approach, we assume that the plan is error free and focus on causes of plan execution failures.

Kalech and Kaminka [11, 12] apply *social diagnosis* in order to find the cause of an anomalous plan execution. 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 its observations, beliefs and role, each agent chooses 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).

Carver and Lesser [3], and Horling et al. [10] 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. Their approach to diagnosis concentrates on specifying the exact causes of the failing of one single *component* (task) of a plan. Diagnosis is based on observations of a component without taking into account the consequences of failures of such a component with respect to the remaining plan. In our approach, instead, we are interested in methods to detect plan execution failures considering the whole plan.

Witteveen and Roos [18, 14, 15] show how classical MBD can be applied to plan execution. Their method is especially suited to identify failing plan steps based on two or more partial observations of a plan execution. They show that an important class of diagnoses, namely minimal maximal-informative diagnoses, can be identified in polynomial time. de Jonge et al. [9, 8] extended the approach of Witteveen and Roos introducing discrete event systems to model the unforeseen changes of state variables representing plan steps, agents, equipment and objects in the environment. Although many types of plan execution failures can be identified, violations of temporal constraints cannot.

de Jonge et al. [5, 7] present an approach to plan diagnosis closely related to the approach proposed in this paper. de Jonge et al. apply discrete event systems to describe linear plans of individual agents. However, temporal aspects of the agents' plan cannot be modeled. The agents' plans may interact through constraints over the states of plan steps (representing for instance resource constraints). Diagnosis is subsequently applied to identify disruption events causing constraint violations. If disruption events are observable, future constraint violations can be predicted and diagnosis is used to propose repair events to avoid these future constraint violations. This paper extends the approach of de Jonge et al. by enabling diagnosis of temporal constraint violations.

Discrete Event Systems Discrete Event Systems are a modeling method of real-world systems based on finite state machines (FSM) [4]. 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 [16]. 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, the finite state machines modeling individual components have to be combined into a global finite state machine [17]. Unfortunately the number of states of the global model grows exponentially in the number of components, making it infeasible for modeling large systems. Therefore, in recent work, instead of a single finite state machine, multiple finite state machines that interact by exchanging events have been proposed. Here, a state transition of a FSM may may trigger new events causing state transitions of other FSMs. Methods for diagnosing these DESs have been proposed in the literature [1, 13].

3 Modeling plan execution

This section formalizes the description of the plan execution to be diagnosed. First, we specify the DES that forms the basic component for modeling plan execution and the sequences of events that drive the system of coupled DESs. Subsequently, we specify how a plan and its environment are modeled using DESs.

The Discrete Event System We assume a set of objects $\mathcal{O}$ which can be partitioned into plan steps $\mathcal{O}^{plan}$, agents $\mathcal{O}^{ag}$, equipment $\mathcal{O}^{eq}$, environment objects $\mathcal{O}^{env}$ and constraint objects $\mathcal{O}^{constr}$. Each object $o \in \mathcal{O}$ is described by a Discrete Event System [4]. Unlike the DES specified in [6] which is based on timed automaton, this DES is based on a standard finite state machine. Note that we will make a distinction between *events* and *event types*. We introduce this distinction in order to explicit distinguish different occurrences of the same "event".

Definition 1 *An object $o \in \mathcal{O}$ is a Discrete Event System $(S_o, s_o, E_o^{in}, E_o^{out}, \tau_o)$ where*

- *S_o is a set of states,*
- *$s_o \in S_o$ is the initial state,*
- *E_o^{in} is the set of event types the object may react to,*
- *E_o^{out} is the set of event types the object may generate, and*
- *τ_o is a set of state transition rules that are instances of $(S_o \times E_o^{in} \cup \{\text{-}\} \times E_o^{out} \times S_o)$. Here, '-' is a reserved event type symbol denoting the absence of an event. A rule in τ_o will be denoted by $s \xrightarrow{[\varepsilon/\varepsilon']} s'$ where $s \in S_o$ is the state before the transition, $s' \in S_o$ is the resulting state, $\varepsilon \in E_o^{in}$ denotes that an event of type ε triggers the transition, and $\varepsilon' \in E_o^{out}$ denotes that an event of type ε' is generated by the transition.*

Events and event types We make a distinction between events and event types in order distinguish different occurrences of the same "event". For instance, during a day at an airport there can be several 'no-show' events, some of which may cause a delayed departure of a plane. In order to identify the specific *instance* of the 'no-show' events that cause a delayed departure, we make a distinction between an event type ε and an event e which is a specific instance of the event type: $type(e) = \varepsilon$.

Because of the distinction we make between event types and events, we can order the latter. We use $e < e'$ to denote that an event e *causally* occurs before an event e'. The adjective '*causally*' denotes there must also be a causal dependence between the two events.

The order in which events occur is important for making diagnoses. Although some approaches assume that only the order of the events on individual DESs (the objects) can be known [13], here we will not make this assumption. Because of the time scale in air traffic control and because clocks can be synchronized through for instance GPS, we assume a total order of events, which we denote by the *temporal ordering* relation $<$ over events. We will use a sequences of events $\Pi = \langle e_1, ..., e_n \rangle$ to describe a set of events ordered with respect to $<$. Clearly, $e_i < e_j$ iff $i < j$. We use $\overline{\Pi} = \{e_1, ..., e_n\}$ to denote the set of events used to construct the sequence Π, and $\Pi \restriction E$ denotes the restriction of the sequence Π to those events occurring in the set E. The subsequence relation $\Pi \sqsubseteq \Pi'$ is defined as Π being the sequence we get by eliminating events from Π': $\Pi \sqsubseteq \Pi'$ iff $\Pi = (\Pi' \restriction \overline{\Pi})$. Moreover, $\Pi - \Pi'$ denotes the removal of the subsequence $(\Pi' \restriction \overline{\Pi})$ from Π, and $\Pi \sqcup \Pi'$ denotes the creation of a sequence respecting $<$ such that $\Pi \sqsubseteq (\Pi \sqcup \Pi')$, $\Pi' \sqsubseteq (\Pi \sqcup \Pi')$ and $(\overline{\Pi} \cup \overline{\Pi'}) = \overline{(\Pi \sqcup \Pi')}$.

Any model has a boundary of what is and what is not modeled. For instance, we may model the state of the weather but not the processes that determine the changes of the weather. Still, based on the weather forecasts we wish to be able to adapt the state of the weather object. External events can be used for this purpose. An external event is an input event of an object that is not generated by some object; i.e., it is not an output event of some object and there is no rule generating it. So, the set of corresponding event types is defined as:

$$E^{ext} = \bigcup_{o \in \mathcal{O}} E_o^{in} - \bigcup_{o \in \mathcal{O}} E_o^{out}$$

Unlike some formalizations of DESs, we do not explicitly partition the set of event types $E_o = E_o^{in} \cup E_o^{out}$ of an object into observable and unobservable events. In the intended application domain, observability of an event may depend on the circumstances.

The environment In a domain such as air traffic control, the environment has an important influence on the intended execution of a plan. Unforseen changes in the state of the environment such as snow on runways or strong headwinds, may influence the temporal execution of a plan. In order to identify this type of influences using diagnosis we first need to model them.

The deviations in the strength of the aircraft's headwind is a continuous function over time. Obviously, we will never have enough information to identify this function in our diagnostic process. Therefore, we should abstract from the continuous function and use abstract values such as *strong-tailwind*, *tailwind*, *no-wind*, *headwind* and *strong-headwind* instead. These values hold for certain time intervals. Hence, we define the environment by a set of objects $\mathcal{O}^{env}$ where with each object $o \in \mathcal{O}^{env}$ a finite set of possible states S_o is associated.

A state change of one object may cause a state change in another object. For instance heavy snowfall may influence the state of the runway on an airport. If a state change of, for instance, the weather immediately influences the state of the runway, coupled Discrete Event Systems (DES) would be an obvious choice to describe the causal dependencies between objects. However, heavy snow fall does not immediately cause the closure of a runway. Therefore, an event should be generated after an object has been in a certain state for some period of time. Using such a 'time delayed' event we may change the state of the runway after it has been snowing for a specified period of time. Time delayed events do not solve the problem completely. How long it will have to snow before snowfall changes the state of the runway, may depend on the condition of the runway such as the presence of salt, and on other events such as snow removal activities. Therefore, in this example the time delayed event must be generated by runway and not by the weather. Figure 1 gives an illustration.

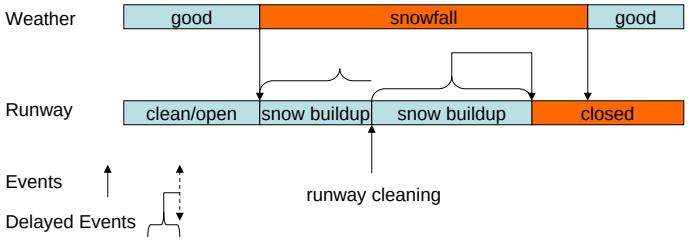

Figure 1: An application of delayed events

Note that the event generated by the weather causes the state change of the runway from 'clean/open' to 'snow buildup'. The event 'runway cleaning' indicating the removal of snow from the runway, causes a state change from 'snow buildup' to 'snow buildup'. It implies that the runway cleaning event resets the timer for the period snow is building up on the runway.

In [6], we specified an interval in which the event causing a transition to the state 'closed' should occur. Abstracting from the time at which an event should occur enables us to describe the state transition of the runway by:

$$\text{snow buildup} \xrightarrow{[\text{-/closing}]} \text{closed}$$

This transition indicates that the state of the runway *may* change from 'snow buildup' to 'closed' without any event causing the transition. During the transition a 'closing' event is generated. This event may be used by other DESs and may be used to observe the state change of the runway.

Plans Since the execution of a step (action) of a plan may depend on the state of the environment objects and since the state of environment objects may change during the execution, we cannot use a representation in which plan steps are treated as atoms. Therefore most representations are not suited for our purpose. What we need is a representation that enables us to state that a plan step finishes too late because it started too late or because a delay occurred during execution as a result of unforseen changes in the environment. Hence, we should be able to assign states to plan steps and these states may change during the execution of a plan step. This suggests that we should also use Discrete Event Systems to model plan steps [7]. We will therefore use a special set of objects $\mathcal{O}^{plan}$ to represent plan steps. Figure 2 illustrates a DES model of a plan step $p \in \mathcal{O}^{plan}$, consisting of:

- a special *initial* state representing the expected initial situation before the execution of the plan step,
- a set of inactive states representing the environment changes that may influence the execution of the plan step,

- a set startup states representing (*i*) that some but not all preceding plan steps have finished as well as how they have finished, and (*ii*) the environment changes that may influence the execution of the plan step,
- a set of active states representing how the plan step is executed, and
- a set of finish states of the plan step.

Note that no startup states are needed in a linear plan.

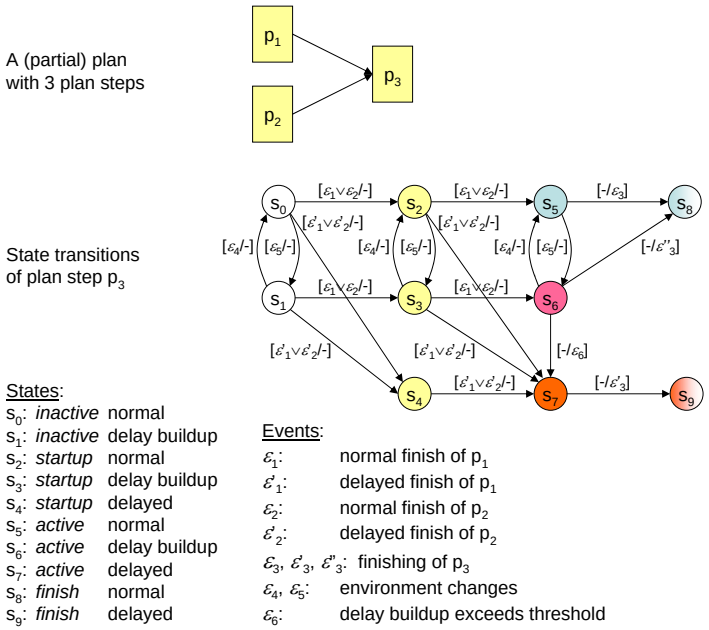

Figure 2: Modeling a plan step of a plan.

Based on the state of a plan step p, at a certain time point a transition from an active state to a finish state must occur. This transition depends on the time a plan step has been in a certain active state. Since we abstract from time information, no event is needed to trigger the transition. Hence we can formulate the following transition rules for the finite state machine in figure2.

$$s_5 \xrightarrow{[-/\varepsilon_3]} s_8, \quad s_6 \xrightarrow{[-/\varepsilon''_3]} s_8, \quad s_7 \xrightarrow{[-/\varepsilon'_3]} s_9$$

Note that the events of type $\varepsilon_3, \varepsilon'_3, \varepsilon''_3$ are generated in order to signal the end of the plan step. They can for instance be used to activate the next plan step of the plan. Also note that these transitions may only occur after all occurrences of the events of types $\varepsilon_4, \varepsilon_5$ and ε_6. Approaches for diagnosis of DESs (e.g. [1, 13]) can handle this issue.

Constraints A plan description normally consists of a set of plan steps with precedence relations between the plan steps. We can distinguish two types of precedence relations, namely those that describe the order of plan steps needed to guarantee the desired effect of a plan and precedence relations that have been added to avoid resource conflicts. Although all precedence relations can be modeled using the above described events, we will use a different description for precedence relations that are added for avoiding resource conflicts. When, for instance, an aircraft is delayed, it may be better to change the planned landing sequence of aircraft so that other aircraft can still arrive as scheduled. Therefore, we will model resource constraints separately as constraints over the states of plan steps requiring the same resource. These constraints can be used to specify for instance that a combination of plane a being delayed and plane b being early is not allowed because both planes are scheduled on the same gate.

Constraints specify combinations of states of objects that are (dis)allowed. Violation of some constraints is a reason for diagnosis. Since approaches for distributed diagnosis of coupled DESs determine diagnoses that explain observed events, we should model each constraint by a DES that generates a *constraint violation event* when a disallowed combination of states of other DESs occurs. By using the events that are generated by the state transitions of an object, the DES of the object $o \in \mathcal{O}^{cstr}$ representing the constraint can generate events indicating a constraint violations.

The model Using the discrete event systems introduced in this section and applying the modeling method discussed, we can now build a model of the possible ways of executing a plan. To this model, we add a set of expected external events. These events specify how the environment is expected to evolve. For instance, describing the expected weather changes.

To summarize, we have to:

- describe the environment objects $\mathcal{O}^{env}$, their state transitions, their states at time point 0, and the rules generating events;
- describe the states of each plan step $\mathcal{O}^{plan}$, their state transitions, states of the plan steps at time point 0, and, using information about the schedule of each plan step, the rules generating (finishing) events.
- if necessary, also describe the equipment $\mathcal{O}^{eq}$ and the agents $\mathcal{O}^{ag}$; and
- describe the constraints $\mathcal{O}^{cstr}$ that must hold between states of different agents' plan steps;
- describe the set of *expected* external event sequences Π^{exp}.

This gives us the execution model $M = (\mathcal{O}, \Pi^{exp})$ with $\mathcal{O} = \mathcal{O}^{env} \cup \mathcal{O}^{plan} \cup \mathcal{O}^{eq} \cup \mathcal{O}^{ag} \cup \mathcal{O}^{cstr}$.

4 Diagnosis & explanation

A diagnosis of a plan execution is defined as well as an explanation of observed constraint violations.

Diagnosis The goal of diagnosis given a plan diagnosis problem (M, Π^{obs}) consisting of the execution model $M = (\mathcal{O}, \Pi^{exp})$ and a sequence of observed events (including constraint violations), is to identify the external events that explain the observed events Π^{obs}. Note that, in the literature, one often aims at a unique identification of failure events [16]. However, the described techniques can also provide multiple explanations in the absence of observations leading to a unique diagnosis.

Definition 2 *Let (M, Π^{obs}) be a plan diagnosis problem and let $M = (\mathcal{O}, \Pi^{exp})$ be a model of the intended plan execution. Moreover, let a sequence of external event Δ be a* candidate diagnosis*.*

Δ is a diagnosis *of a plan execution diagnosis problem iff there exists a sequence Π specifying the execution history of the coupled DESs such that $\Delta = \Pi \restriction E^{ext}$, $\Pi^{obs} \sqsubseteq \Pi$ and $(\Pi^{exp} \restriction \overline{\Pi}) \sqsubseteq \Delta$.*

Preference criteria There may be several diagnoses Δ according to definition 2. The quality of these diagnoses need not be the same. Preference criteria are used to select a subset of the diagnoses, usually, the most probable diagnoses. A criterium that is often used for diagnoses is preferring diagnoses that minimize the difference with the normal state of affairs. In plan diagnosis this would be the difference with the expected external events sequence Π^{exp}.

The difference between Π^{exp} and a diagnosis Δ consists of two aspects, (i) the unexpected events that occurred according to the diagnosis, and (ii) the expected timed events that did not occur according to the diagnosis. Assuming that the a priori probability that Δ differs from Π^{exp} is sufficiently small, *we prefer diagnoses Δ that minimize the differences between Δ and Π^{exp}*.

A difficulty in comparing Π^{exp} and Δ is that there are transitions to the same state s starting from different initial states triggered by event of different types. For instance, two event types for a transition to a state representing strong winds, one if there was no wind and one if there was a light breeze. If the expected light breeze did not occur, we should still be able to infer that the change to strong winds did occur. Therefore we will restrict the external events to so called *absolute* events. An absolute event causes a transition to a new state independent of the previous state thereby simplifying comparison of external events.

Definition 3 *An external event type $\varepsilon \in E^{ext}$ of an object $o \in \mathcal{O}$ is an* absolute event *iff*
for every $s \xrightarrow{[\varepsilon/\varepsilon']} s'', s' \xrightarrow{[\varepsilon/\varepsilon'']} s^* \in \tau$: $\varepsilon' = \varepsilon''$ and $s'' = s^*$.

Explanation In our application domain of air traffic control one often claims that during normal daily operation all relevant events, including the external events, are observable. This does not imply that no constraint violation will occur when agents execute their plans. On the contrary, air traffic controllers are working around the clock to avoid incidents. Clearly, if all external events that have occurred, are observed, then $\Delta = \Pi^{obs} \restriction E^{ext}$ is a diagnosis. However, such a diagnosis does not give an adequate explanation of an observed constraint violation.

For the purpose of plan repair, distributing cost of a plan repair, improvements of future plans, and so on and so forth, we would like to know which external events are accountable of the observed constraint violations. A diagnosis does not provide this information. A diagnosis specifies the external events that explain the observed events. From such a diagnosis we can determine the unexpected external events that occurred and the expected external events that did not occur without linking the presence or absence of specific events to specific constraint violations. A specific constraint violation is caused by the presence or absence of specific external events. For instance, the absence of expected tail wind can be the cause of a plane's late arrival. An *explanation* identifies these unexpected external events as well as unexpectedly absent external events *causing* the constraint violation.[2]

Determining an explanation for a constraint violation is not straight forward because deriving a constraint violation from external events is a non-monotonic process. To illustrate this, consider an aircraft that has a delayed departure, but still arrives on time at its destination because of strong tail winds. However, because no gate is available after landing, the aircraft has a delayed arrival resulting in a constraint violation. The constraint violation could be explained by considering the external event causing the delayed departure while ignoring in the explanation the external events causing the tail wind and the unavailability of a gate after landing. Clearly, this is not a proper explanation of the delayed arrival because the plane landed on time. Also a set containing all external events that influences the schedule of the aircraft does not give a proper explanation. The only events that give a proper explanation of the constraint violation in this example are those causing the unavailability of a gate.

How do we determine the external events that explain an event e generated by a constraint object; i.e., the events causing a constraint violation? First, observe that an explanation must enable us to generate the event e to be explained. Since an explanation consists of unexpected external events that occurred and of expected external events that did not occur, the result of removing the latter events from the sequence of expected events Π^{exp} and subsequently adding the former events to it, must enable us to predict the event e to be explained. As was illustrated by the example above, we must also ensure that every larger explanation also enables us to predict the event e.

Definition 4 *Let $M = (\mathcal{O}, \Pi^{exp})$ be a model of the plan execution and let Δ be a diagnosis. Moreover, let e be an observed event for which we seek an explanation and let $\langle e \rangle$ represent the corresponding sequence of observations.*

An explanation is a couple (X^p, X^a) with $X^p \sqsubseteq (\Delta - \Pi^{exp})$ and $X^a \sqsubseteq (\Pi^{exp} - \Delta)$ such that:

- $\Upsilon = (\Pi^{exp} - X^a) \sqcup X^p$ *is a diagnosis of the sequence of observations: $\langle e \rangle$;*
- *for no $(\mathcal{Y}^p, \mathcal{Y}^a)$ with $X^p \sqsubseteq Y^p \sqsubseteq (\Delta - \Pi^{exp})$ and $X^a \sqsubseteq Y^a \sqsubseteq (\Pi^{exp} \ominus \Delta)$:*
  $\Xi = (\Pi^{exp} - Y^a) \sqcup Y^p$ is not *a diagnosis of the sequence of observations: $\langle e \rangle$.*

Note that in the above definition, the first item gives an 'argument' for (X^p, X^a) being an explanation while the second item states that there is no counter-argument.

5 Example

This section illustrates the relevance of the model in our application domain, the field of air traffic control, using a small example.

At Schiphol airport, flight KL1243 to DeGaulle Paris, which is docked at gate E11, is delayed because it has to wait for passengers (the expected off-block event after which the aircraft is to taxi to the runway does not occur at the planned time, but occurs 15 minutes later). After further investigation it becomes apparent that the passengers that KL1243 is waiting for are transfers from flight D845. Flight D845, from Heathrow London, was delayed due to strong headwinds, and only just began de-boarding at gate D21.

Figure 3 shows the model of the schedule for flights KL1243 and D845. As can be seen, the expected in-block time of D845, which is the start of the de-boarding, was expected before the off-block time of KL1243, but due to (unexpected) weather changes has been delayed. This delay causes a longer boarding time of KL1243, which is noted by the delay in the occurrence of it's off-block event.

Clearly, the diagnosis will contain the explanation: 'strong head winds: London to Amsterdam'. Note that this diagnosis can be used to predict that other flights from the same direction will probably be delayed as well (until the weather changes).

[2]Here, we use a pragmatic interpretation of the concept 'causes'.

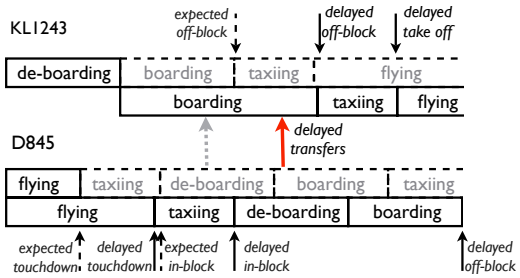

Figure 3: Example diagnosis.

6 Conclusion

Identifying causes of temporal constraint violations is an important problem in our application domain of air traffic control as well as many other domains. Information about the causes may support plan repair and can help improving new plans. In this paper we have investigated whether we can model a plan execution for the purpose of diagnosis using a Discrete Event Systems without considering explicit time information. We have shown that a diagnosis of temporal constraint violations can be made using such a model and that causes of individual constraint violations can be identified.

Future work will focuses on efficient distributed implementations of the model. We will investigate extensions of the approaches proposed in [7] and in [13].

References

[1] P. Baroni, G. Lamperti, P. Poglianob, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence*, (110):135–183, 1999.

[2] L. Birnbaum, G. Collins, M. Freed, and B. Krulwich. Model-based diagnosis of planning failures. In *AAAI 90*, pages 318–323, 1990.

[3] N. Carver and V. Lesser. 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, 2003.

[4] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[5] F. de Jonge and N. Roos. Plan-execution health repair in a multi-agent system. In *PlanSIG 2004*, 2004.

[6] F. de Jonge, N. Roos, and H. Aldewereld. Using DESs for temporal diagnosis of multi-agent plan execution. In *MATES 2007*, 2007.

[7] F. de Jonge, N. Roos, and H. van den Herik. Keeping plan execution healthy. In *Multi-Agent Systems and Applications IV: CEEMAS 2005, LNCS 3690*, pages 377–387, 2005.

[8] F. de Jonge, N. Roos, and C. Witteveen. Diagnosis of multi-agent plan execution. In *Multiagent System Technologies: MATES 2006, LNCS 4196*, pages 86–97, 2006.

[9] F. de Jonge, N. Roos, and C. Witteveen. Primary and secondary plan diagnosis. In *The International Workshop on Principles of Diagnosis, DX-06*, 2006.

[10] B. Horling, B. Benyo, and V. Lesser. Using Self-Diagnosis to Adapt Organizational Structures. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 529–536. ACM Press, 2001.

[11] M. Kalech and G. A. Kaminka. On the design of social diagnosis algorithms for multi-agent teams. In *IJCAI-03*, pages 370–375, 2003.

[12] M. Kalech and G. A. Kaminka. Diagnosing a team of agents: Scaling-up. In *AAMAS 2005*, pages 249–255, 2005.

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

[14] N. Roos and C. Witteveen. Diagnosis of plans and agents. In *Multi-Agent Systems and Applications IV: CEEMAS 2005, LNCS 3690*, pages 357–366, 2005.

[15] N. Roos and C. Witteveen. Models and methods for plan diagnosis. In *Formal Approaches to Multi-Agent Systems (FAMAS'06)*, 2006.

[16] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosibility of discrete event systems. *IEEE Transactions on Automatic Control*, 40:1555–1575, 1995.

[17] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology*, 4:105–124, 1996.

[18] C. Witteveen, N. Roos, R. van der Krogt, and M. de Weerdt. Diagnosis of single and multi-agent plans. In *AAMAS 2005*, pages 805–812, 2005.