Noname manuscript No.
(will be inserted by the editor)

Diagnosis from Scenarios, and applications to security

Loïc Hélouët · Hervé Marchand · Blaise Genest · Thomas Gazagnaire

the date of receipt and acceptance should be inserted later

Abstract Diagnosis of a system consists in providing explanations to a supervisor from a partial observation of the system and a model of possible executions. This paper proposes a partial order diagnosis algorithm that recovers sets of scenarios which correspond to a given observation. The main difficulty is that some actions of the monitored system are unobservable but may still induce some causal ordering among observed events. We first give an offline centralized diagnosis algorithm, then we discuss a decentralized version of this algorithm. We then give an online diagnosis algorithm, and define syntactic criteria under which this algorithm can run with finite memory. The last contribution of the paper is an application of diagnosis techniques to a security problem called *anomaly detection*. Anomaly detection consists in comparing what occurs in the system with usual/expected behaviors, and raising an alarm when some unusual behavior (meaning a potential attack) occurs.

1 Introduction

Complexity of distributed systems calls for automated techniques to help designers and supervisors in their tasks. Before correcting a system's software, or taking a decision (for instance a reconfiguration of a network), stakeholders need to obtain information on what occurred in a network before entering a faulty configuration, on the current state of the system, etc. The role of diagnosis is to provide a feedback to supervisors of a system (this can be online, to obtain some information on the current status of a running system, or offline,

Work partially supported by the RNRT POLITESS

L. Hélouët · H. Marchand
INRIA, Centre Rennes - Bretagne Atlantique, Campus de Beaulieu, RENNES, France

B. Genest
IRISA/CNRS, Campus de Beaulieu, RENNES, France

T. Gazagnaire
INRIA Sophia Antipolis - Méditerranée, Sophia Antipolis,France

to know why a fault occurred and then correct the incriminated part of the system). Usually, diagnosis relies on observation of the system (for instance some information stored in log files during execution), and on some a priori knowledge of the behaviors of a system. However, observations can only be partial: distributed systems are now so complex that monitoring every event in a running system is not realistic. In telecommunication networks, for example, the size of complete logs recorded at runtime grows fast, and can rapidly exceed the storage capacity of any machine, or the computing power needed to analyze them. Furthermore, the time penalty imposed by the observation to the system also advocates for a partial observation. Hence, monitoring a system means choosing an appropriate subset of observable events. This choice is clearly part of the design of a complex system.

Several techniques are frequently called "diagnosis" while addressing different goals. Any kind of technique that provides online or offline information on a system to a supervisor can be called diagnosis, but we will focus more precisely on two of them, namely fault diagnosis, and history diagnosis. *Fault diagnosis* answers the question whether a system is faulty, i.e it has reached some bad configuration or executed some events that should have not occurred. The main challenge in fault diagnosis is to decide whether for given sets of faults and observable events the system is diagnosable, i.e. the occurrence of a fault can eventually be detected after a finite number of observations [23]. Diagnosis is then performed by an observer that monitors observable actions and raises an alarm when needed.

History diagnosis reconstructs an actual set of possible executions of a system from a partial observation. The a priori knowledge on the system available for this is defined as a model of system's behaviors. The objective is then to build a set of plausible explanations (runs of the model) that comply with the observations [4]. Then, these potential explanations can be exhaustively checked to find a fault, or to provide feedback to system's supervisor. Note that Fault and History diagnosis try to solve different problems: fault diagnosis tries to infer if a fault has occurred (and very often which fault), while history diagnosis may provide several explanations (faulty or not) for a given observation.

Within this paper, we will address history diagnosis of distributed systems. The major objective of this work is to exploit concurrency in the system, and avoid combinatorial explosion using partial order models. It is well-known that interleaved models can be of size exponentially greater than concurrent models. Hence, as long as an analysis of a system does not need to study all global states, true concurrency models should provide efficient solutions. In this paper, we propose to model the diagnosed system with High-level Message Sequence Charts (or HMSCs for short), a scenario formalism [11]. The observation of the system is provided as a partial order, and the explanation is given as a non-interleaved representation of all possible executions that may have generated the observation according to the model.

The authors of [4] already address history diagnosis with a partial order model (safe Petri nets). In this approach, diagnosis is an incremental construction of an unfolding of the net model. The incremental aspect of this approach is clearly well adapted for online diagnosis, but does not allow for a compact representation of explanations. When unobservable events can be iterated an unbounded number of times, this incremental approach becomes impossible (unfolding may never stop).

This paper proposes a history diagnosis technique based on partial order models. We consider that observations of a monitored system are provided as a partial order, and use a scenario language called High-level Message Sequence Charts (HMSC) as a model of the observed system to find explanations of an observation. The algorithm detailed in this paper starts from an observation O given as a partial order, an HMSC model H of the possible behaviors of the system, and the knowledge of the type of events that have been recorded in O to build a product between O and H that contains all possible explanations.

We do not impose restrictions on the observation architecture: observed events occurrence may be collected in a centralized way, or separately by distributed observers. However, we will consider that for a given process, all observed events are totally ordered. Furthermore, the processes may be equipped to record the respective order between events located on different processes (this ordering can be deduced for example from messages numbering, or from a vectorial clock tagging observed events [16,7]). Hence the observation O may specify some particular ordering between events that is not only induced by emissions and receptions of messages. This additional information can be used to refine the set of explanations provided by the model. Indeed, if an event e happens before an event e' in the observation, then in any possible explanation provided by the model, e must be causally related to e'. We also assume that the observation mechanisms inserted in the distributed system are lossless. That is, if an observable event does not appear in the observation, then we have the information that it did not occur.

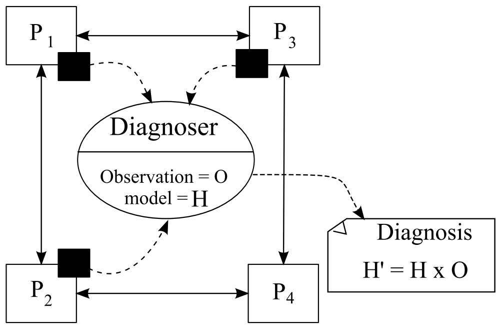


Fig. 1 Scenario-based diagnosis framework

Figure 1 shows an usual diagnosis architecture. The monitored system is composed of 4 processes P_1, P_2, P_3 and P_4, represented by white squares. Communications between these processes are symbolized by arrows between processes. Some sites in the system are equipped by sensors or software probes, represented by dark squares in the figure, that detect the occurrence of some events (a message is sent or received, a timeout has occurred, a program has reached a specified point in its control flow, ...). These events are sent to a centralized mechanism, the *diagnoser*. The communications to the diagnoser can use the communication means of the monitored system, or another network dedicated to this task. We only suppose that no observed event is lost, and that all messages that are sent from a process to the diagnoser respect a FIFO ordering (this can be easily achieved by tagging the information sent with the clock of the monitored process). Only a subset of everything that occurs on a process is monitored. The diagnoser uses a model (in our case a HMSC H) that describes all possible behaviors of the system (or at least a reasonably large subset of them), and builds an observation from the set of all events that it receives. The role of the diagnoser is to output a new model (the *diagnosis*) that defines the set of all executions of the model that are consistent with the observation (the *explanations*). In our case, the output of the diagnoser is a new HMSC that describes all possible explanations of an observation. Note that so far, we make no assumption on whether the diagnosis algorithm is implemented in a centralized or decentralized way.

– The first result of this paper is that we can finitely represent the set of runs of a distributed system that explains a particular observation O. The explanation produced is a generator of all executions of our model for which the projection on observed events is compatible with O. More precisely, we show that the set of explanations can be described by another HMSC. This gives the basis of a centralized diagnosis algorithm.
– The second result of this paper is that a global explanation can be reconstructed from local diagnosis performed for each pair of process. This allows for an easy partition of the diagnosis problem into smaller tasks, and hence for a distribution of a global diagnosis task to several diagnosers. Within this setting, each diagnoser computes separately the set of executions that can explain what it has observed. At the end of all local diagnosis, a last step combines the local explanations to produce a global explanation.
– The third part of this paper focuses on an online version of the algorithms described in the previous sections. In this case, the diagnoser receives observed events one after another, and builds a diagnosis incrementally. In this algorithm, the main objective is to reuse at step n the part of the diagnosis computed at state $n-1$. We show an efficient algorithm that allows for this incremental construction. We also show that under some sufficient syntactic restrictions on the model, the online diagnoser can work with finite memory.
– The fourth part of this paper shows an interesting application of diagnosis to security, and more precisely to *anomaly detection*. An anomaly is an unexpected behavior, which can mean that an attack of a system has occurred. We propose a definition of anomaly detection as a weakened version of diagnosis, namely the diagnosis existence problem. When no explanation of an observation can be found in a model, then the observed behavior is considered as illegal. This diagnosis-based anomaly monitoring reuses the diagnosis algorithms, but for different purposes. Anomaly monitoring can then be performed offline or online.

This paper is an extended version of a preliminary work published in [10]. It is organized as follows: section 2 introduces the scenario language used, and section 3 introduces the formal definition of observations and explanations. Section 4 defines the main algorithms for diagnosis, and shows how to retrieve explanations in a decentralized framework. Section 5 show how to adapt offline algorithms to perform online detection. Section 6 shows the application of diagnosis to anomaly detection. Section 7 concludes this work.

2 Scenarios

Scenarios are a popular formalism to define use cases of distributed systems. Several languages have been proposed [11,21], but they are all based on similar representations of distributed executions with compositions of partial orders. We use Message Sequence Charts, a scenario language standardized by ITU [11]. A part of this standard is well accepted and used in the industry. The advantage of working with partial order models is well known: as soon as a problem can be solved without enumerating all global states of a model, the solution can be computed more efficiently (saving up to exponential time) [1]. Formally, Message Sequence Charts can be defined by two layers of description. At the first level, basic MSCs (or simply MSCs in the rest of the paper) describe asynchronous interactions among components of a system called *instances*. An instance usually represents a process, or a group of processes of a distributed system. For simplicity, we will consider that instances in MSCs represent processes of a system, and we will indifferently use one term or the other. In a MSC, instances exchange messages (in asynchronous mode), and can also perform atomic actions. These interactions are then composed by High-Level Message Sequence Charts, a finite-state automaton labeled by MSCs. We can define these two models as follows:

First of all, we will consider executions of a system composed of a set of processes $\mathcal{P}$, executing actions from an alphabet Σ which contains labels of the form $p(a)$ denoting internal actions, $p!q(m)$ denoting sending of message m from process p to process q, and $q?p(m)$ denoting the reception of a message m on p by q. We can partition Σ into $\Sigma_! \cup \Sigma_? \cup \Sigma_a$, respectively the sending, reception and internal actions, or into $\bigcup_{p\in\mathcal{P}} \Sigma_p$, where Σ_p denotes the actions executed by process p.

Definition 1 *A* (basic) Message Sequence Chart *(MSC for short) over a set of processes $\mathcal{P}$ and a set of actions Σ is a tuple $M = (E, \leq, \alpha, \mu, \phi)$, where:*

- *E is a set of events*
- *$\leq$ is a partial order relation (reflexive, transitive, antisymmetric) over E*
- *$\alpha : E \longrightarrow \Sigma$ is a labeling function. As for labels, we can partition E into $E_! = \alpha^{-1}(\Sigma_!)$, $E_? = \alpha^{-1}(\Sigma_?)$ and $E_a = \alpha^{-1}(\Sigma_a)$.*
- *$\mu : E_! \longrightarrow E_?$ is a bijection pairing message emissions and receptions.*

[1] Note however that some problems such as model-checking of LTL formulae, etc usually rely on a computation of global state space. Hence, some problems can not be addressed within their partial order representations.

– *$\phi : E \longrightarrow \mathcal{P}$ is a function that associates a process to each event. For a given event $e \in E$, $\phi(e)$ will be sometimes called the* locality *of e. We can define a partition $\{E_p\}_{p\in\mathcal{P}}$ of the set of events according to the processes of M, i.e. $\forall p \in \mathcal{P}, E_p = \phi^{-1}(p)$.*

Furthermore, all MSCs must satisfy the following properties:

i) *$\forall p \in P, \ \leq \cap E_p \times E_p$ is a total order. We will often denote by $\leq_p$ the restriction of $\leq$ to events located on process p and by $<$ the intransitive and irreflexive reduction of $\leq$.*
ii) $(\mu \cup \bigcup_{p\in\mathcal{P}} \leq_p)^* \ = \ \leq$

For a more detailed description of all MSC features, we refer interested readers to [11]. To an MSC, one can make easily correspond a graphical representation which is close to their formal definition, as the visual aspect of a MSC is almost the Hasse diagram of the underlying partial order defined by processes and messages. The only difference between MSCs and Hasse diagrams occurs when two messages between the same processes are crossing: the Hasse diagram shows a total order on sending and receiving events, while the corresponding MSC depicts both messages. Note that map μ is a bijection, that is every message sent in a MSC is also received within this MSC (and conversely). We will then say that MSCs are *communication closed*.

Example 1 Figure 2 represents a Message Sequence Chart: three processes called $Sender$, $Medium$ and $Receiver$ exchange asynchronous messages $Data$, $Info$ and Ack, and process $Sender$ performs local action a.

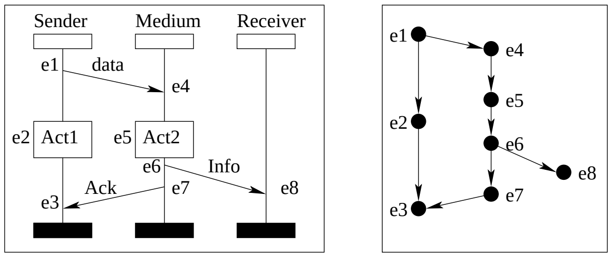


Fig. 2 A MSC Example and its underlying partial order

Instances are symbolized by a vertical line enclosed between a white and a black rectangle. Messages are symbolized by arrows from the emitting instance to the receiving one. Atomic actions are symbolized by a rectangle enclosing the name of the action. In this example, we get for example {Sender!Medium(data), Medium?Sender(data)} $\subseteq \Sigma$. *We also have* $\alpha(e_1)$ =Sender!Medium(data), $\alpha(e_8)$ = Receiver?Medium(Info), *etc. We get* $E_{\text{Sender}} = \{e_1, e_2, e_3\}$ *and* $\mu(e_1) = e_4$, $\mu(e_7) = e_3$.

A *linear extension* of a MSC M is a sequence of events $w = e_1 \ldots e_{|E|}$ such that each event of E appears exactly once in w, and $\forall i, k \in 1..|E|$, $e_{i+k} \not\leq e_i$. Back to the previous example, a possible linear extension of M is $e_1e_4e_2e_5e_6e_7e_8e_3$.

A *linearization* of a MSC M is a word $\sigma = a_1 \ldots a_{|E|}$ of Σ^* that is the labeling of some linear extension w of M i.e. $\sigma = \alpha(w)$ (with α extended from letters to words). The set of all linearizations of a MSC M is denoted $Lin(M)$. Computing the linearizations of an MSC resumes to providing an interleaved interpretation of its partial order. This calculus should be avoided when possible, as the automaton recognizing $Lin(M)$ can be exponential in the size of M.

From now on, we will consider that all MSCs are defined on similar set of processes $\mathcal{P}$, even if these processes are not active in the MSC. We will also denote by M_ϵ the empty scenario. MSCs alone do not have enough expressive power to describe complex behaviors. They can only define a single and finite partial ordering among events. However, the MSC formalism has been extended with several operators to allow iterations, alternatives, and sequential composition. *Sequential composition* allows to glue two MSCs along their common instance axes to build larger executions. It is formally defined as follows:

Definition 2 *Let M_1, M_2 be two MSCs. The* sequential composition *of M_1 and M_2 is denoted $M_1 \circ M_2$, and is the MSC $M_1 \circ M_2 = (E_1 \uplus E_2, \leq_{1\circ 2}, \alpha_1 \uplus \alpha_2, \mu_1 \uplus \mu_2, \phi_1 \uplus \phi_2)$, where $\leq_{1\circ 2} = (\leq_1 \cup \leq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid \phi(e_1) = \phi(e_2)\})^*$, with $\uplus$ denoting disjoint union, and $f_1 \uplus f_2$ denotes a function defined over $Dom(f_1) \uplus Dom(f_2)$, that associates $f_1(x)$ to any $x \in Dom(f_1)$ and $f_2(x)$ to any $x \in Dom(f_2)$.*

Note that sequential composition does not impose synchronization among instances: events of M_1 and M_2 can still be concurrent if they are located on distinct processes.

Example 2 Figure 3 shows an example of sequential composition of two MSCs. In the composition $M1 \circ M2$, action a and the sending of message m, for example, are still concurrent events.

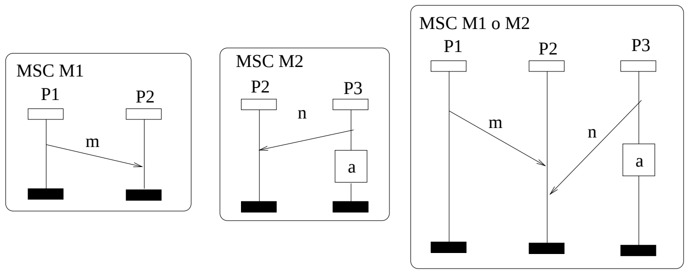


Fig. 3 Sequential composition of MSCs

Sequential composition of MSCs allows for the definition of infinite sets of MSCs of arbitrary size. However, the usual way to define such sets of MSCs and to give a more intuitive structure to a set of MSCs is to use a higher level formalism called High-level Message Sequence Charts (HMSC) that proposes several other operators such as alternative and iteration. A HMSC can be seen as an automaton labeled by MSCs. More formally, a HMSC can be described as follows:

Definition 3 *A High-level Message Sequence Chart (or HMSC for short) is a tuple $H = (N, \longrightarrow, n^i, \mathcal{M}, F)$ where:*

- *N is a set of nodes, $F \subseteq N$ is a set of final (or accepting) nodes, $n^i \in N$ is an initial node,*
- *$\mathcal{M}$ is a finite set of MSCs,*
- *$\longrightarrow \subseteq N \times \mathcal{M} \times N$ is a transition relation.*

In a HMSC, nodes define potential global states of the system, that are used to glue MSCs. Note however that these nodes do not impose any synchronization among processes, and that a system may never pass through these global states. A HMSC is then just the generator for a set of partial orders.

Example 3 Figure 4 contains an example of a HMSC H. The initial node n_0 is connected to a downward triangle, and the only final node n_1 is connected to an upward triangle. The transitions of H are (n_0, M_1, n_0) and (n_0, M_2, n_1). The HMSC of Figure 4 describes a simple interaction between a client and a server. The client can query a server and has to wait for an answer before doing anything else (MSC $M1$). After answering, the server stores the question in its database. This behavior can be iterated an arbitrary number of times (described by the loop in the HMSC graph) before the client closes the session (MSC $M2$). Note that the atomic action Record can be concurrent with the reception of an answer, but also with the sending of message Close by the client.

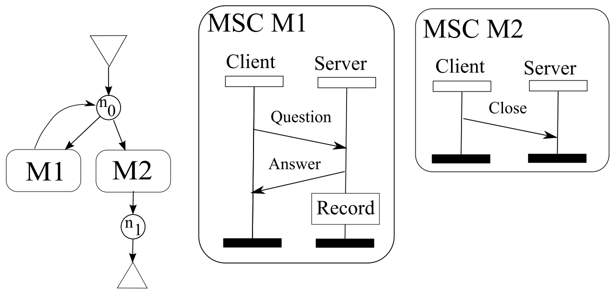


Fig. 4 High-level MSC

The whole terminology for automata applies to HMSCs. A *path* of H is a sequence of transitions $\rho = n_0 \xrightarrow{M_1} n_1 \ldots \xrightarrow{M_k} n_k$ such that $\forall i \in 0...k-1, (n_i, M_{i+1}, n_{i+1}) \in \longrightarrow$. If $n_0 = n^i$, then ρ is called an *initial path*, and if $n_k = n_0$, then ρ is called a *cycle*. If $n_k \in F$, ρ is called an *accepting path*. A HMSC H hence defines a set of accepting paths denoted by $\mathbb{P}_H$. To each path ρ of $\mathbb{P}_H$, one can associate a unique MSC M_ρ obtained by concatenation of successive MSCs labeling transitions of ρ, i.e. $M_\rho = M_1 \circ \cdots \circ M_k$. The MSC M_ρ will be called a *run* of H. A HMSC can then be considered as the generator for the set of MSCs $\mathcal{F}_H = \{M_\rho \mid \rho \in \mathbb{P}_H\}$, or for the set of linearizations $\mathcal{L}_H = \bigcup\limits_{M \in \mathcal{F}_H} Lin(M)$.

Remark 1 Even if H is an automaton, its linearization language $\mathcal{L}_H$ may not be a regular language. It has been shown that High-level Message Sequence Charts embed the expressive power of Mazurkiewicz traces [18], and rational relations. Consequently, several classical model-checking problems (language inclusion, equality, universality, vacuity of intersection,...) become undecidable for HMSCs. It is even undecidable in general to know whether the linearization language of a HMSC is regular. Fortunately, several subclasses of HMSC allow for the decision of some problems. For instance, regular HMSCs [1] allow for all model-checking problems that are feasible on automata, globally cooperative HMSCs [9] allow for decision procedures for inclusion, equality, and vacuity of intersection, etc.

3 Observation

Let us now define the essential notions that will be used to find explanations of an observation. An observation O performed during an execution of a system should be an abstraction of an existing execution (i.e. an abstraction of a MSC). We will suppose that on each instance of our distributed system, a subset of events is monitored: every time a monitored event e is executed, a message is sent by a local observer to the supervision mechanism. In the following, we will only suppose that observations are **lossless** (all events that are monitored are effectively reported when they occur), **faithful** (observers never send events that did not occur to the supervising architecture, and do not create false causalities), and received within a **bounded delay** t_{obs}. The set of types of monitored events is defined as an observation alphabet Σ_{obs}. The observations can contain additional ordering information (built from local observations and additional information such as packet numbers, vectorial clocks,...), and are thus considered as labeled partial orders. We will also consider that for a given instance, the observation is a sequence, that is, the communication between local observers and the supervision architecture is FIFO. Note also that events are not observed on all instances, hence we define a set $\mathcal{P}_{obs} \subseteq \mathcal{P}$ on which events are monitored (i.e. $\mathcal{P}_{obs} = \phi(\alpha^{-1}(\Sigma_{obs}))$). Formally, observations can be defined as labeled partial orders as follows.

Definition 4 *An* observation *is a tuple $O = (E_O, \leq_O, \alpha_O, \mu_O, \phi_0)$, where $E_O, \leq_O, \alpha_O, \phi_O$ have the usual meaning in MSCs and μ_O is a partial application that pairs events of E_O. Observations have to define a total order $\leq_{O_p}$ on each process, as for MSCs, and satisfy the inclusion $(\mu_O \cup \bigcup_{p\in\mathcal{P}} <_{O_p})^* \subseteq \leq_O$.*

From this definition, observations are a relaxed version of MSCs with less constraints on ordering: they are not necessarily communication-closed, as some emissions of $E_{O_!}$ may not have an image through μ_O, and some receptions may not be the image of an emission. This is justified by the fact that we do not want to enforce that both the sending and reception of messages are observed. Furthermore, condition *ii*) of MSCs is relaxed, that is we do not require anymore that the union of local ordering and message mapping forms a transitive reduction of $\leq_O$ as in MSCs. Indeed, all events are observed locally, and nothing guarantees that message sendings and receptions are correctly mapped, nor that both ends of a message are observed. However, vectorial stamping, packet numbering or similar information exchanged among processes can help building a causal order that is richer than a simple collection of sequences of observed events on each process. Observations can be composed like MSCs using the $\circ$ operator. In the sequel, we will adopt the following graphical convention for observations. Processes will be represented as in MSCs, but without the black rectangle ending the process line. Events will be represented as boxes labeled by the event type, and the covering of the ordering relation will be depicted as arrows between causally related events.

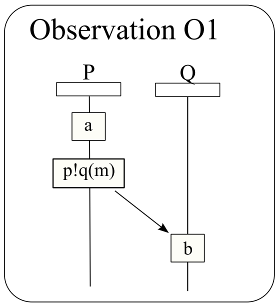


Fig. 5 An example observation

Example 4 Figure 5 shows an example of observation. Processes P and Q are monitored. In this observation, two events have occurred on P: an atomic action a and the sending of a message m to Q. A single event has occurred on Q, an atomic action b. Note that the message sending precedes action b.

For an arbitrary partial order $O = (E_O, \leq_O, \alpha_O, \mu_O)$, we will denote by $<_O$ the covering of relation $\leq_O$, i.e $x <_O y$ iff $\nexists z \in E_O \setminus \{x, y\}, x \leq_O z \leq_O y$. For a given event $e \in E_O$, we will denote by $\downarrow(e)$ the set of all causal predecessors of e. We will also denote by $max_{\leq}(p)$ the maximal event located on process p. Furthermore, slightly abusing the notation, for a set of events E, we will denote by $O \setminus E$ the restriction of O to $E_O \setminus E$, and write $e \in O$ instead of $e \in E_O$. Finally, we say that a set of event $E \subseteq E_O$ is a *prefix* of O if for all $a \leq_O b$ with $b \in E$, then $a \in E$.

Now that we have defined the observations that are produced by the probes and collected by our diagnosis architecture, let us show how MSCs and HMSCs can be used to explain observation.

Definition 5 *Let $M = (E, \leq, \alpha, \mu, \phi)$ be a MSC over a set of processes $\mathcal{P}$ and a set of actions Σ. Let $\Sigma_{obs} \subseteq \Sigma$ be an observation alphabet. The* projection *of M over Σ_{obs} is an observation denoted by $\Pi_{\Sigma_{obs}}(M) = (E_O, \leq_O, \alpha_O, \mu_O, \phi_O)$ such that $E_O = E \cap \alpha^{-1}(\Sigma_{obs})$, $\leq_O = \leq \ \cap \ (E_O \times E_O)$, and α_O (resp. μ_O, ϕ_O) is the restriction of α (resp. μ, phi) to E_O.*

Note that an MSC is also an observation (but the converse is not true). However, what a monitoring system observes from these executions are just projections. Indeed, it is not possible to instrument a system in such a way that any instruction or event occurring on every process is recorded. This also holds for the causal relationships between observed events. Hence, the observed order among observed events might be smaller than the actual causal ordering of the execution. This is captured by the notion of *sub-order* defined below. Clearly, this means that observations are sub-orders of projections of executions on observed events. Furthermore, as the systems described are supposed to be networks of machines communicating asynchronously, the integration of observation mechanisms should follow the same requirements. Some events are then collected on each site, and sent to an observer. These observations are also sent asynchronously, which means that when performing online diagnosis (and thus anomaly detection), we have to take into account that the observation available so far might be extended in the future after reception of additional observed events. This is captured by the notion of *prefix*.

Definition 6 *Let $O = (E_O, \leq_O, \alpha_O, \mu_O, \phi_O)$ be an observation. A* prefix *of O is an observation $O' = (E'_O, \leq'_O, \alpha'_O, \mu'_O, \phi'_O)$ such that $E'_O \subseteq E_O$ is a prefix of O, and $\leq'_O, \alpha'_O, \mu'_O, \phi'_O$ are restrictions of $\leq_O, \alpha_O, \mu_O, \phi_O$ to E'_O. A* sub-order *of O is an observation $O' = (E_O, \leq'_O, \alpha_O, \mu'_O, \phi'_O)$ such that $\leq'_O \subseteq \leq_O$ and $\mu'_O \subseteq \mu_O$.*

We will say that an MSC M and an observation O are consistent when the observation is something that might have been collected during the execution of M. This can be defined by a *matching relation* between O and M:

Definition 7 *Let $O = (E_O, \leq_O, \alpha_O, \mu_O, \phi_0)$ be an observation, and $M = (E, \leq, \alpha, \mu, \phi)$ be a MSC over a set of processes $\mathcal{P}$ and a set of actions Σ. O* matches *M with respect to an observation alphabet Σ_{obs} (denoted by $O \rhd_{\Sigma_{obs}} M$) if and only if O is a prefix of a sub-order of $\Pi_{\Sigma_{obs}}(M)$.*

Whenever $O \rhd_{\Sigma_{obs}} M$, we will say that M is an explanation of O (w.r.t observation alphabet Σ_{obs}). Let us detail this definition. We require O to be a sub-order of a prefix of the projection of M. The prefix requirement imposes that when an event is observed in O, all the observable preceding events on the same process have also been observed. Note however that M can still contain observable events that have *not yet* been observed. The sub-order requirement imposes that any causal ordering found in O is actually an ordering described in M (but the converse needs not hold). Note that this matching definition is close to the definition of matching proposed by [19,17] for verification purpose.

Proposition 1 *$O \rhd_{\Sigma_{obs}} M$ if and only if there exists a matching function $h_{O,M} : E_O \longrightarrow E_M$ that sends events of E_O onto events of M such that:*

- *$h_{O,M}$ respects the labeling ($\alpha(h_{O,M}(e)) = \alpha(e)$) and causal ordering of O ($e \leq_O f \Longrightarrow h_{O,M}(e) \leq_M h_{O,M}(f)$)*
- *for every pair of events $e \leq_M f$ in M located* **on the same process***, and such such that $\alpha(e) \in \Sigma_{obs}$ and $\alpha(f) \in \Sigma_{obs}$, $h_{O,M}(f)$ defined implies that $h_{O,M}(e)$ is also defined. Furthermore, this function is unique.*

Proof: It is easy to see that if $h_{O,M}$ exists, then it is the (unique) function $h_{O,M} : E_O \to E_M$ that sends the k-th event of E_O on instance p onto the k-th event of $\pi_{\Sigma_{obs}}(M)$ on instance p for all $k \in \mathbb{N}$ and $p \in \mathcal{P}_{obs}$ (due to the fact that O is totally ordered on each process, and must be closed by precedence). If $h_{O,M}$ does not preserve causal ordering for some events located on distinct processes, then O can not be a sub-order of a prefix of M. Conversely, if O is a sub-order of a prefix of M, then for every ordered pair of events $e \leq_O f$ in O, we have $h_{O,M}(e) \leq_M h_{O,M}(f)$. □

Corollary 1 *Given an observation O such that $O \rhd_{\Sigma_{obs}} M$ and an observation O' such that*

- *O' is a prefix of O,or*
- *O' is a sub-order of O,or*

then $O' \rhd_{\Sigma_{obs}} M$. Furthermore, if $O' = \Pi_{\Sigma'}(O)$ for some alphabet $\Sigma' \subseteq \Sigma_{obs}$, then $O' \rhd_{\Sigma'} M$

Proof: The proof is straightforward, as it is sufficient to consider the restriction of $h_{O,M}$ to events of O' to obtain a matching relation from O' to M. □

Proposition 2 *Let O be an observation and M be a MSC. Then, checking whether $O \rhd_{\Sigma_{obs}} M$ can be done in $O(|M| + | \leq_O |.| \leq_M |)$.*

Proof: The first step to verify a matching relation is to build the mapping $h_{O,M}$ from O to M, that is compare sequences of observable events along each process. This can be computed in linear time in the size of M. Then, for each pair of events (a, b) appearing in $\leq_O$ we have to verify that $h_{O,M}(a) \leq_M h_{O,M}(b)$. □

Example 5 Let us illustrate matching on the examples of Figure 6, where $\Sigma_{obs} = \{a, b\}$, O_1, O_2, O_3, O_4 are observations, M_1, M_2, M_3, M_4 are MSCs, and the matching relation h_{O_i,M_i} that sends an observation onto an execution is represented by dotted arrows when it exists.

- *Let us consider O_1 and M_1: there is an injective mapping from the observation to a prefix of the explanation. a's and b are concurrent in the observation, but the order O_1 can clearly be injected in M_1, hence $O_1 \rhd_{\Sigma_{obs}} M_1$.*
- *For the pair O_2, M_2, there is also an injective mapping that maps O_2 to a prefix of the projection of M_2 onto Σ_{obs}. c does not have to be matched, as it is not an observed event.*
- *For the pair O_3, M_3, a and b are unordered in the explanation M_3 and hence the observation O_3 can not be injected in M_3.*
- *For the pair O_4, M_4, there is no injective mapping satisfying the three conditions of the morphism defined in proposition 1. Indeed, an occurrence of b should have been observed between two a's. Hence, M_4 is not an explanation of O_4.*

From these examples, one may notice that the observation of some events provides information on whether a peculiar execution M is an explanation of what has been observed, but also that the causal ordering of some events in the observation or their absence can also be used to rule out some possible explanations (this is the case for the pairs (O_3, M_3) and (O_4, M_4)).

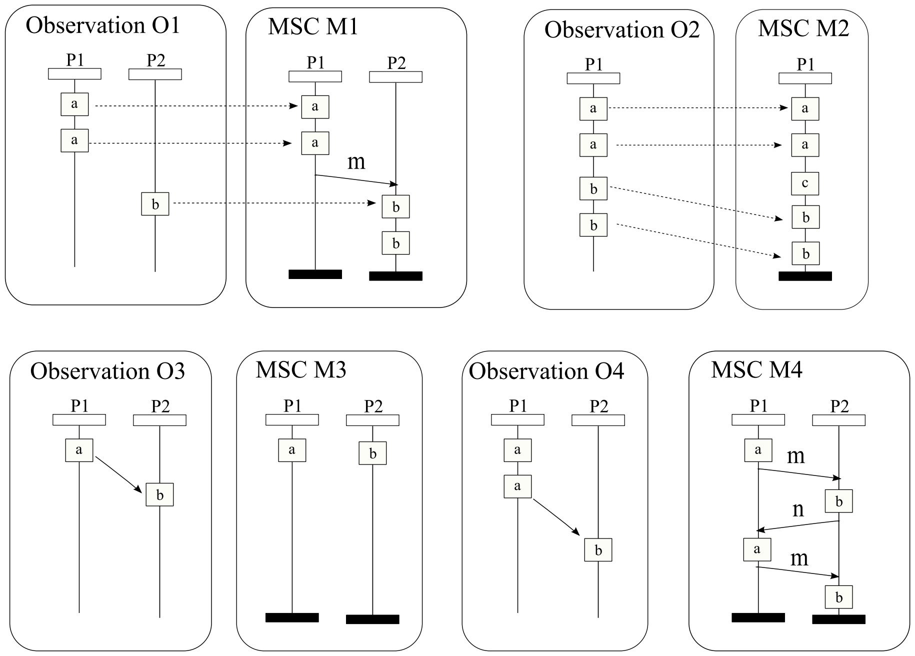


Fig. 6 Two matching examples w.r.t $\{a, b\}$ and two counter examples

Definition 8 *Let O be a partial order over Σ_{obs} and H be an HMSC. The set of* explanations provided by *H for an observation O is the set of paths $\mathbb{P}_{O,H} \subseteq \mathbb{P}_H$ such that $\forall \rho \in \mathbb{P}_{O,H}$, $O \rhd_{\Sigma_{obs}} M_\rho$.*

Notice that the set of explanations provided by H is not always finite nor its linearization language is regular, but we will prove that it can be described by an HMSC in section 4, theorem 1. As already mentioned, observations may be collected either in a centralized or a distributed way, and observed events can be sent to supervising mechanisms via asynchronous communications. Hence, the model of our system can describe runs which projections are all larger than the observation collected so far. Note however that thanks to the prefix condition, our framework does not impose observations to be **complete** projections of an MSC labeling an accepting path of H, but should only embed into an explanation.

Our goal is to build incrementally the set of all explanations provided by a HMSC. This means that if we study MSC concatenations, we should be able to test whether it is worth or not continuing along a path of a HMSC. This leads us to introduce the notion of *compatibility* defined as follows:

Definition 9 *A MSC M is* compatible *with an observation O if and only if there exists a MSC M' such that $O \rhd_{\Sigma_{obs}} M \circ M'$. Given a HMSC H and a run $\rho \in \mathbb{P}_H$, then M_ρ is* compatible *with an observation O (w.r.t HMSC H) if and only if there exists ρ' such that $\rho\rho' \in \mathbb{P}_H$ and $O \rhd_{\Sigma_{obs}} M_{\rho\rho'}$.*

When H is clear from the context, we will drop the reference to H and simply write that M_ρ is compatible with O, or even ρ is compatible with O. It is worthwhile noting that when M and O are compatible, then there exists a unique maximal embedding function h that sends a prefix of O onto events of M, and such that any embedding h' of O into $M \circ M'$ is an extension of h. Hence, the unique embedding h of O into some MSC in $\mathcal{F}_H$ can be built incrementally.

4 Offline diagnosis

The main objective of our diagnosis approach is to extract from an HMSC H a generator for the set of explanations $\mathbb{P}_{O,H}$ of an observation O. This generator can be defined as a product between the HMSC and the observation, with synchronization on monitored events. Hence, we will build a new automaton whose nodes are product of a node of the original HMSC with the subset of events of O observed so far, that will be called the *progress* of the observation. For instance, a path leading to the product state (v, E_O) should generate an execution that embeds O.

Next we outline some difficulties we will face up in order to build this automaton. First, we can remark that it is not possible to say that that a path $\rho = n_0 \xrightarrow{M_1} n_1 \ldots \xrightarrow{M_k} n_k$ is not an explanation of O just by considering the projections $\Pi_{\Sigma_{obs}}(M_1), \ldots, \Pi_{\Sigma_{obs}}(M_k)$. This is basically due to the fact that in general $\Pi_{\Sigma_{obs}}(M_1 \circ M_2) \neq \Pi_{\Sigma_{obs}}(M_1) \circ \Pi_{\Sigma_{obs}}(M_2)$: the former may provide more ordering on projected events than the latter (see for instance the MSCs M_1 and M_2 in Figure 7). For similar reasons, we cannot use as a basis for diagnosis a copy of the original HMSC which transitions are labeled by projections of MSCs as shown by Example 6.

Second, one of the main difficulty is to know the influence of unobservable events and of concatenation on the causal ordering of observable events. As already mentioned, valid explanations may contain an arbitrary number of unobserved events. Fortunately, we can always keep an abstract and bounded representation of these unbounded orders, by projecting runs of our models on observable events, and recalling some causalities. This abstraction of runs will be modeled by a partial function $g_{O,M} : \mathcal{P} \longrightarrow 2^O$ that associates to each instance $p \in \mathcal{P}$ the observed events of O preceding the last event (observed or not) on instance p after playing some run M of an HMSC. More formally, for an observation O and a MSC M compatible with O, we have

$$g_{O,M}(p) = h^{-1}_{O,M}\Big(\downarrow\big(max_{\leq_M}(p)\big)\Big) \cap Dom(h_{O,M}),$$

where $h_{O,M}$ is the (unique) maximal embedding of prefixes of O in M. Notice that function $g_{O,M}$ defines an abstraction of a run that is not redundant with the order contained in O, since the run of the HMSC represented by $g_{O,M}$ can provide more ordering on observed events than O.

Example 6 Let us illustrate the use of function g with an example. Consider the two MSCs of figure 7, and the observation alphabet $\Sigma_{obs} = \{a, b, b', c, c'\}$.

O_1 and O_2 are the the projections of M_1 and M_2 on Σ_{obs}. We can remark that $\Pi_{\Sigma_{obs}}(M_1 \circ M_2)$ and $O_1 \circ O_2$ comport isomorphic sets of events, but define different causal orderings on theses events (b and c' are causally ordered in $\Pi_{\Sigma_{obs}}(M_1 \circ M_2)$ but not in $O_1 \circ O_2$. The reason is that the causality from $Medium$ to $Receiver$ induced by message $Info$ is lost during projection. Let us suppose that MSC M_1 has been played as an explanation of observation O_1. Then, the function g_{O_1,M_1} computed after M_1 to explain observation O_1 associates event a to process sender, events $\{a, b\}$ to process $Medium$, and events $\{a, b, c\}$ to process $Receiver$.

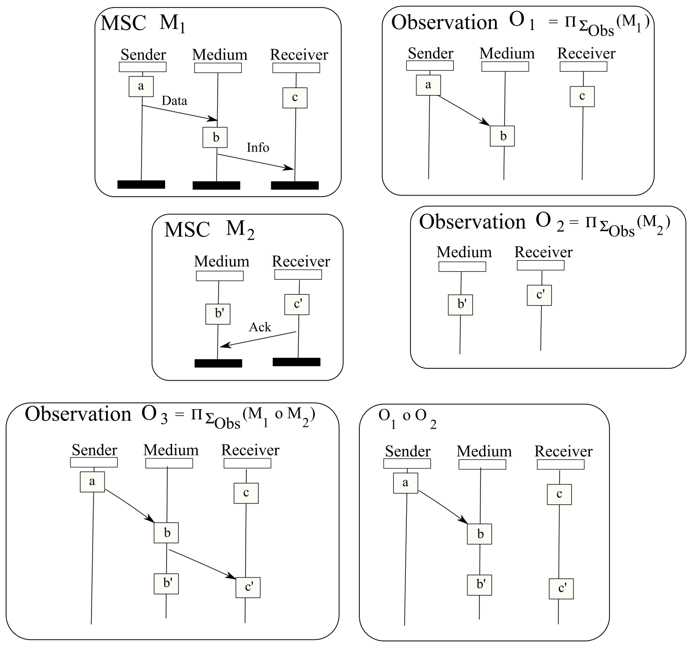

MSC M1
Sender
Medium
Receiver
a
Data
b
Info
c
Observation O1 = ΠΣObs(M1)
MSC M2
b'
c'
Ack
Observation O2 = ΠΣObs(M2)
Observation O3 = ΠΣObs(M1 o M2)
O1 o O2

Fig. 7 Concatenation and Projection

Let us now show that for a given observation O, the projections and the function g can be computed incrementally. To do so, we first show how to compute the projection of $M_1 \circ M_2$ according to the projections of M_1 and M_2 (i.e $\Pi_{\Sigma_{obs}}(M_1)$ and $\Pi_{\Sigma_{obs}}(M_2)$) and the function g_{O,M_1}:

Proposition 3 *Let $M_1 = (E_1, \leq_1, \alpha_1, \mu_1, \phi_1)$, $M_2 = (E_2, \leq_2, \alpha_2, \mu_2, \phi_2)$ be two MSCs, and let $\Pi_{\Sigma_{obs}}(M_1) \circ \Pi_{\Sigma_{obs}}(M_2) = (E, \leq, \alpha, \mu, \phi)$. Then for any observation O such that M_1 is compatible with O,*

$$\Pi_{\Sigma_{obs}}(M_1 \circ M_2) = (E, \leq', \alpha, \mu, \phi), \text{ where}$$

$$\leq' = \Big(\leq \cup \{(x, y) \in \Pi_{\Sigma_{obs}}(M_1) \times \Pi_{\Sigma_{obs}}(M_2) \mid \exists z \leq_2 y, x \in g_{O,M_1}(\phi(z)\ )\} \Big)^*$$

Proof: Let us suppose that there exists (x, y) that are ordered in $\Pi_{\Sigma_{obs}}(M_1 \circ M_2)$, but not in $\leq'$. Then, obviously $x \in E_1$ and $y \in E_2$, and furthermore,

$\phi(x) \neq \phi(y)$. As (x, y) are ordered in $\Pi_{\Sigma_{obs}}(M_1 \circ M_2)$ without being on the same process, then there exists an event $x' \in E_1$ such that $x \leq x'$, and event $y' \in E_2$ such that $\phi(x') = \phi(y')$. Hence, we have that $x \in g(\phi(x'))$, and $x \leq' y$, contradiction. □

The next step is to show that we can compute the function $g_{O,M_1 \circ M_2}$ from g_{O,M_1} and $M2$.

Proposition 4 *Let $M_1 = (E_1, \leq_1, \alpha_1, \mu_1, \phi_1)$, $M_2 = (E_2, \leq_2, \alpha_2, \mu_2, \phi_2)$ be two MSCs such that $M_1 \circ M_2$ is compatible with O. Then, for every $p \in \mathcal{P}$,*

$$\begin{aligned} g_{O,M_1 \circ M_2}(p) = g_{O,M_1}(p) &\cup \{g_{O,M_1}(\phi(e)) \mid e \leq_2 e', \phi(e') = p\} \\ &\cup \{e \in h^{-1}_{O,M_1 \circ M_2}(\pi_{\Sigma_{Obs}}(M_2) \cap O) \mid e \leq_2 e', \phi(e') = p\}, \end{aligned}$$

where $h_{O,M_1 \circ M_2}$ is the largest embedding of prefixes of O into $M_1 \circ M_2$.

Proof: Suppose that there exists a process $p \in \mathcal{P}$ and a event $e \in O$ such that $e \in g_{O,M_1 \circ M_2}(p)$ but e is not in the incremental computation of g. Then, clearly $e \notin g_{O,M_1}(p)$. If $e \in M_1$, then $e \in g_{O,M_1 \circ M_2}(p)$ if and only if there exists a causal chain of events $h_{O,M_1 \circ M_2}(e) < e_1 < ...e_i < e_j < ... < e_n$ in $M_1 \circ M_2$ such that $e_n \in E_2$ and is located on process p, $e_i \in M_1$ and $e_j \in M_2$ are located on the same process. Hence by definition, $e \in g_{O,M_1}(\phi(e_i))$ and also belongs to the incremental construction of $g_{O,M_1 \circ M_2}(p)$. If $e \in M_2$, then $e \in g_{O,M_1 \circ M_2}(p)$ if and only if there exists a causal chain $h_{O,M_1 \circ M_2}(e) < ... < e_n$ in M_2 such that $e_n \in E_2$ is located on process p. This case is also captured by the incremental construction. □

Hence it is sufficient when studying an arbitrary long path ρ of a HMSC that is compatible with O to memorize the finite set of observed events in M_ρ and g_{O,M_ρ} to be able to build incrementally a faithful projection of the MSC labeling any continuation of this path (and make sure that this continuation is still compatible with the observation). We are now ready to build the product $\mathcal{A}_{O,H}$ of an observation O on an alphabet Σ_{obs} and a HMSC H:

Definition 10 *Given a HMSC H and an observation O on an alphabet Σ_{obs}, $\mathcal{A}_{O,H}$ is an automaton defined by: $\mathcal{A}_{O,H} = (Q, \delta, \mathcal{M}, q_0, F')$, where δ is a new transition relation, $Q \subseteq N \times Prefix(O) \times \mathcal{F}$, where $\mathcal{F}$ is the set of functions from $\mathcal{P}$ to 2^O.*

- *$q_0 = (n^i, M_\epsilon, g_\emptyset)$, where $g_\emptyset$ is a function over an empty domain.*
- *$\Big((n, E, g), M, (n', E', g')\Big) \in \delta$ with $E \neq O$ iff*
  - *$n \xrightarrow{M} n'$,*
  - *For every process p, either E''_p is a prefix of E_{Op}, or E_{Op} is a prefix of E''_p, where $E'' = E \uplus \pi_{\Sigma_{Obs}}(M)$. When this property holds, then $E' = E_O \cap h^{-1}(E'')$, where h is the largest partial mapping of events of E_O onto events of E'' that preserves local ordering $\leq_p$, and labeling. Note that E' is necessarily a prefix of O. When the property does not hold for MSC M, then the transition is not allowed.*

  - $g'(p) = g(p) \cup \{g(\phi(e)) \mid e \leq_M e', \phi(e') = p\} \cup \{e \in \pi_{\Sigma_{Obs}}(M) \mid e \leq_M e', \phi(e') = p\}$,
  - *For all $a, b \in E'$ with $a <_0 b$, either $a, b \in E$ (in this case, the ordering of a and b has already been checked in former transitions), or $h(a) \leq_M h(b)$, or $\exists c \leq_M h(b)$ with $a \in g(\phi(c))$ (the existence of a causal ordering between a and b is ensured by proposition 3).*
- $\Big((n, E_O, g), M, (n', E_O, g)\Big) \in \delta$ iff $n \xrightarrow{M} n'$.
- $F' = \{(n, E_O, g) \mid n \in F\}$,

Note that $g(p)$ is updated only when the explanation provided at a given state is incomplete. It is updated to memorize the observable events in the causal past of the last event (observed or not) executed by each instance. Similarly, we make sure during construction of a transition $\Big((n, E, g), M, (n', E', g')\Big) \in \delta$ that any order $a <_O b$ is preserved in E': either $h(a), h(b)$ are predecessors of all events of M and their ordering was already checked, or they are ordered in M, or $h(a)$ is in M and $h(b)$ is a successor of an event that necessarily occurs after $h(a)$. We hence ensure by construction that for any path ρ of $\mathcal{A}_{O,H}$, M_ρ is compatible with O. Transitions of $\mathcal{A}_{O,H}$ of the form $\Big((n, E, g), M, (n', E', g')\Big) \in \delta$ can be projected to obtain transitions of the form (n, M, n') that are used in H to move from one state to another. We denote by $\mathbb{L}_{\mathcal{A}_{O,H}}$ the set of accepting paths of $\mathcal{A}_{O,H}$, and by $\mathbb{L}_{H,\mathcal{A}_{O,H}} \subseteq \mathbb{P}_H$ the set of paths of H that are projections of $\mathbb{L}_{\mathcal{A}_{O,H}}$ on the first component of each state.

Theorem 1 *[10] Let $\mathcal{A}_{O,H}$ be the HMSC computed from O and H, and $\rho \in \mathbb{P}_H$. Then $O \rhd_{\Sigma_{obs}} M_\rho$ iff $\rho \in \mathbb{L}_{H,\mathcal{A}_{O,H}}$. Moreover, $\mathcal{A}_{O,H}$ is of size $O(|H| \times |O|^{|\mathcal{P}| \times |\mathcal{P}_{Obs}|})$.*

Proof: It is obvious from the construction of δ that any accepting path ρ of $\mathbb{L}_{H,\mathcal{A}_{O,H}}$ generates a MSC M_ρ such that $O \rhd_{\Sigma_{obs}} M_\rho$, as we forbid any transition where $a <_O b$ and $h_{O,M_\rho}(a) \not\leq_{M_\rho} h_{O,M_\rho}(b)$. Reciprocally, consider $\rho \notin \mathbb{L}_{H,\mathcal{A}_{O,H}}$, then either $\rho \notin H$ and we are done or $\rho \in H$ but $\nexists \rho' \in \mathbb{L}_{H,\mathcal{A}_{O,H}}$ such that ρ is the restriction of ρ' on its first component. Consider ρ_1 and ρ_2 such that $\rho = \rho_1\rho_2$ and ρ_1 is the greatest prefix of ρ that is compatible with a prefix of ρ'. Thus ρ_2 is of the form $(n, M, n').\rho_3$ and $M_{\rho_1} \circ M$ is not compatible with O, which is thus also the case for ρ. □

For the complexity statement, notice that a prefix can be uniquely represented by remembering its last event on each observed instance. Hence the number of prefixes of O is lower than $|O|^{|\mathcal{P}_{Obs}|}$. Moreover, notice that g associates to every instance i a prefix of O. At last, notice that for every state (n, E, g), we have $E = \bigcup_{p \in \mathcal{P}} g(p)$, hence E is superfluous as it can be computed from g. We however kept the prefixes of the observation in the definition of states for the sake of readability of the construction of $\mathcal{A}_{O,H}$. □

Theorem 1 means in particular that $\mathbb{L}_{H,\mathcal{A}_{O,H}} = \mathbb{P}_{O,H}$. Hence, $\mathcal{A}_{O,H}$ is the generator of all explanations of observation O provided by the HMSC model H.

The restriction of $\mathcal{A}_{O,H}$ to coaccessible states of F' is the *diagnosis* provided for observation O from the HMSC model H.

Remark 2 Note however that paths in $\mathbb{L}_{H,\mathcal{A}_{O,H}}$ are not the *minimal paths* embedding O, as $\mathcal{A}_{O,H}$ allows **any** transition of H from its accepting states. To consider only **minimal** paths, one should consider only the relation $\delta' = \delta \cap \{\big((n,E,g),M,(n',E',M')\big) \mid E \neq E_O\}$, and the set of accepting nodes $F' = \{(n,E_O,g)\}$. For a centralized offline diagnosis performed with a complete observation, this has no importance. However, we will see in section 4.2 that when the diagnosis problem is split into sub-problems, it is important to return **all** paths embedding the observation.

Example 7 Consider the HMSC H and the observation O of Figure 8. The HMSC describes the behavior of three processes $P1, P2, P3$. Let us denote by e_1 the occurrence of action a in O and by e_2 the occurrence of action b. Let us

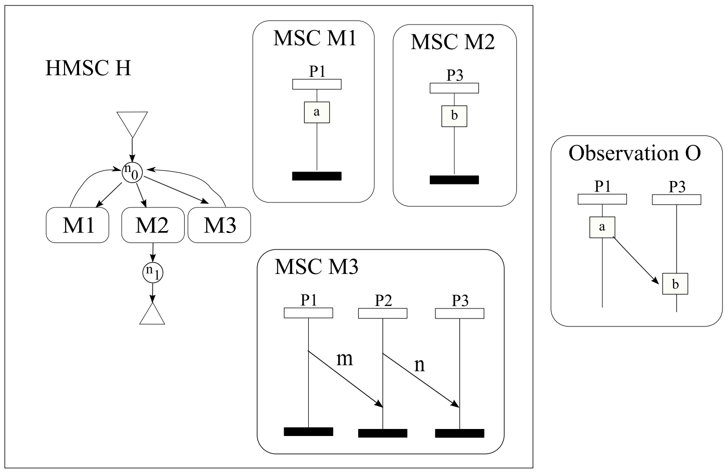


Fig. 8 A HMSC example and an observation

suppose that we have equipped a distributed system to observe any occurrence of actions a and b and that we obtain the observation O. Clearly, $n_0 \xrightarrow{M1} n_0 \xrightarrow{M2} n_1$ is not an explanation of O for the observation alphabet $\Sigma_{obs} = \{a,b\}$, as a and b are not causally related in $M1 \circ M2$. The automaton $\mathcal{A}_{O,H}$ computed from O and H with this observation alphabet is given in Figure 9. The transitions with a dark cross symbolize transitions of the original HMSC that cannot be fired in the diagnosis automaton. For example, from the initial state, the transition labeled by $M2$ cannot be used, as any path ρ starting with this transition would not allow a matching from O to M_ρ. One can easily verify that O matches any MSC composition of the form $M3^* \circ M1 \circ M3 \circ M3 \circ M3^* \circ M2$. Note that if we choose as observation alphabet $\Sigma_{obs} = \{a,b,!m\}$, the observation O has no explanation in H.

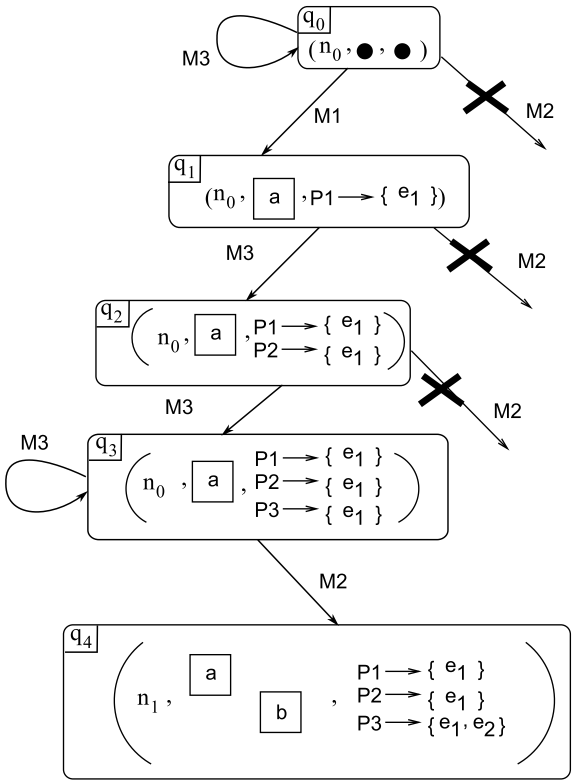


Fig. 9 A diagnosis automaton

4.1 Offline Existence

The diagnosis problem can be simplified to answer a simpler question : is there an explanation for an observation O in H? In the sequel, we will refer to this question as the *existence problem*, which can be formalized as follows: given an HMSC H, an observation alphabet Σ_{Obs} and an observation O, $\exists ? \rho \in \mathbb{L}_{H,\mathcal{A}_{O,H}}, O \triangleright M_\rho$.

Theorem 2 *Let H be a HMSC, Σ_{obs} be an observation alphabet, and O be an observation. Deciding whether there exists an explanation for O in H w.r.t. Σ_{obs} is an NP-complete problem.*

Proof: First, let us show that the existence problem is in NP. There exists an explanation for O in H if and only if we can exhibit a path ρ of H such that $O \triangleright_{\Sigma_{obs}} M_\rho$. Let us suppose that ρ is a path of length greater than $|O|^2.|\mathcal{P}|.|H|$. Whenever, ρ is an explanation for O, this path has at most $|O|$ transitions labeled by an MSC which contains an event e that is the image of some event of O via the matching function h_{O,M_ρ}. Hence, we can exhibit a sub-sequence of consecutive transitions in ρ of size greater than $|O|.|\mathcal{P}|.|H|$ that are only labeled by unobservable MSCs, or which labeling MSCs are not used to explain

O (that is they comport only events which are not in $h_{O,M_\rho}(E_O)$). Then, two cases can appear. Either ρ' is a suffix (resp. a prefix) of ρ, or not. If ρ' is a suffix (resp. a prefix), then we can remove it from ρ to obtain a smaller explanation. If not, then ρ is of the form $\rho = \rho_1.\rho'.\rho_2$. As ρ' is of size greater than $|O|.|\mathcal{P}|.|H|$, it necessarily contains at least $|O|.|\mathcal{P}|$ cycles of H, and hence it is a sequence of transitions of the form $\rho' = u_1.\beta_1.u_2.\beta_2 \dots \beta_{|O|.|\mathcal{P}|}.u_{|O|.|\mathcal{P}|+1}$, where each β_i is a cycle of H. Note that each path ρ corresponds to a path of $\mathcal{A}_{O,H}$, but that loops of H are not necessarily loops of $\mathcal{A}_{O,H}$, as nodes of $\mathcal{A}_{O,H}$ contain a reference to an HMSC node, plus a function g (observed events sets are redundant with g, as shown in theorem 1, and can be forgotten). Hence, each β_i is a cycle from a node n_i to a node n_i in H, and is mapped to a path from a node (n_i, E, g_i) to a node (n_i, E, g_{i+1}) in $\mathcal{A}_{O,H}$. If $g_i = g_{i+1}$, then this path of $\mathcal{A}_{O,H}$ is a cycle, and can be removed from ρ to obtain a smaller explanation $\rho_1.u_1\beta_1 \dots \beta_{i-1}.u_i.u_{i+1}.\beta_{i+1} \dots u_{|O|.|\mathcal{P}|+1}.\rho_2$. If $g_i \neq g_{i+1}$, then there exists at least one process such that $|g_i(p)| > |g_{i+1}(p)|$. Let us suppose that for every $i \in 1..|O|.|\mathcal{P}|-1$, we have that β_i is mapped to a path of $\mathcal{A}_{O,H}$ from (n_i, E, g_i) to (n_i, E, g_{i+1}) with $g_i \neq g_{i+1}$. Then, we necessarily have $g_{|O|.|\mathcal{P}|-1}(p) = E$ for every p, and $\beta_{|O|.|\mathcal{P}|}$ is mapped to a loop of $\mathcal{A}_{O,H}$, and can be removed from ρ to obtain a smaller explanation $\rho_1.u_1\beta_1 \dots \beta_{|O|.|\mathcal{P}|}.u_{|O|.|\mathcal{P}|+1}.\rho_2$. Hence, if an explanation exists for an observation O, then there is necessarily an explanation of length at most $|O|^2.|\mathcal{P}|.|H|$.

Now let us show that for a path $\rho = n_0 \xrightarrow{M_1} n_1 \dots \xrightarrow{M_k} n_k$, we can check in polynomial time whether $O \rhd_{\Sigma_{obs}} M_\rho$. First, the sequential concatenation of all MSCs to obtain M_ρ can be computed in polynomial time in the size of the path. Let m be the maximal size of the causal ordering relation, and n be the maximal number of events in all MSCs of H. We can use the following algorithm to compute the concatenation $M_1 = (E, \leq, \alpha, \mu, \phi)$ of two MSCs $M_i = (E_i, \leq_i, \alpha_i, \mu_i, \phi_i)$, with n_i events and causal ordering of size m_i, $i \in 1, 2$.

- Compute $E = E_1 \uplus E_2$
- initialize $\leq$ with $\leq_1 \uplus \leq_2$
- for every process $p \in \mathcal{P}$, find the maximal event x on p in M_1 and the minimal event on p in M_2, and add $x \leq y$ to the ordering relation. Then compute the closure : for every $z \leq_1 x$ and every $y \leq_2 z'$, add $z \leq z'$ to the ordering relation.

This gives a complexity of $O(n1+n2+m1+m2+p.(n1.m1+n2.m2+m1.m2))$ for concatenation. Computing M_ρ resumes to k concatenations of MSCs with less than $k.n$ events and a causal ordering relation of size smaller than $(kn)^2$, and can hence be performed in polynomial time, and results in a MSC with at most kn events and a causal ordering relation of size at most $(kn)^2$. Then, from proposition 2, verifying that $O \rhd_{\Sigma_{obs}} M_\rho$ can be done at most in $O(k.n + | \leq_O |.(k.n)^2)$. Hence, we can guess a path of H to explain O in polynomial time, and check in polynomial time whether it is an explanation for O. So, the existence problem is in NP.

Now, let us show that the existence problem is NP-hard. We proceed by reduction from the $3SAT$ problem. Let $\phi = C_1 \wedge \cdots \wedge C_m$ be a conjunctive formula in normal form over n variables $v_1, \ldots, v_n$, with m clauses, and where each clause C_i is of the form $C_i = l_i^1 \vee l_i^2 \vee l_i^3$, and each literal l_i^j refers to a variable in $v_1, \ldots, v_n$ or its negation, that is $l_i^j = v_k$ or $l_i^j = \overline{v_k}$, for some $k \in 1..n$. We can build a HMSC H with $n+3$ nodes, $2n+2$ transitions and 2n+2 MSCs, an observation alphabet Σ_{obs} and an observation O such that ϕ is satisfiable iff O has an explanation in H w.r.t. Σ_{obs}. The observation and the HMSC are defined over a set of processes $\mathcal{P} = \{P_{v_1}, \ldots, P_{v_n}\} \cup \{P_{c_1}, \ldots, P_{c_n}\}$. The observation alphabet Σ_{obs} is composed of $m+1$ letters $\{a_0, a_{c_1}, \ldots, a_{c_m}\}$. The observation O comports one occurrence of each letter in Σ_{obs}, and is such that the occurrence of a_0 is located on process P_{v_1}, the occurrence of each a_{c_i} is located on process P_{c_i} and the event labeled by a_0 causally precedes all other events (see Figure 10). The HMSC H, also depicted in Figure 10, comports a set of nodes $Q = \{q_0, q_e, q_f\} \cup \{q_{v_1}, \ldots, q_{v_n}\}$, and is labeled by MSCs $Start, End$ and $T_{v_1}, \ldots, T_{v_n}, F_{v_1}, \ldots, F_{v_n}$. The MSC $Start$ consists in a single occurrence of action a_0 located on process P_{v_1}. The MSC End consists in one occurrence of action a_{c_i} on each process P_{c_i}, $i \in 1..m$. Each MSC $T_i = X_1 \circ \cdots \circ X_m \circ V_i$ is a concatenation of $m+1$ MSCs. Each X_j is either an MSC that contains a message from P_{v_i} to P_{c_j} if one of the literals of clause C_j is v_i, and is the empty MSC otherwise. MSC V_i is a message from process P_{v_i} to process $P_{v_{i+1}}$ if $i < n$ and the empty MSC otherwise. Each MSC $F_i = Y_1 \circ \cdots \circ Y_m \circ V_i$ is a concatenation of $m+1$ MSCs. Each Y_j is either an MSC that contains a message from P_{v_i} to P_{c_j} if one of the literals of clause C_j is $\overline{v_i}$, and is the empty MSC otherwise. Finally, there is a transition from q_0 to q_{v_1} labeled by $Start$, a transition from q_e to q_f labeled by End, and two transitions from q_{v_i} to $q_{v_{i+1}}$ respectively labeled by T_i and F_i for every

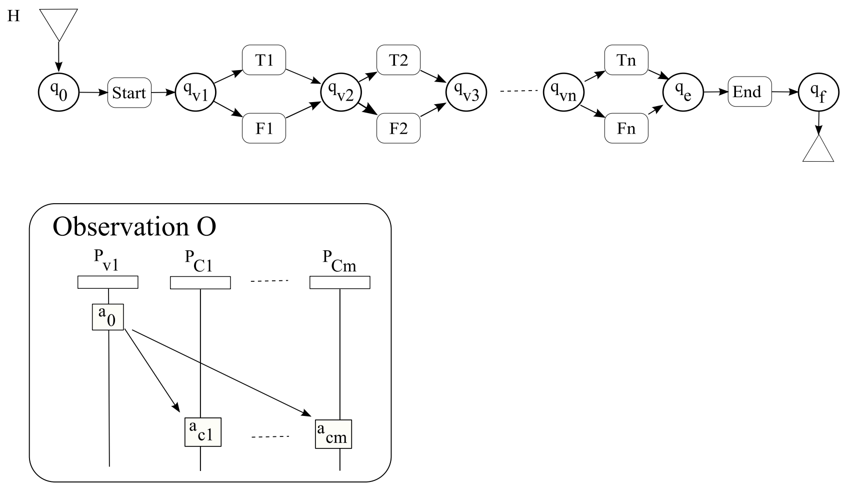


Fig. 10 Encoding SAT problems with an existence problem

$i \in 1..n-1$, and two transitions from q_{v_n} to q_e respectively labeled by T_n and F_n. Clearly, an occurrence of each action in Σ appears in an explanation ρ of O if and only if ρ is a path from q_0 to q_f.

Clearly, ϕ is satisfiable iff there is a variable assignment such that for every clause C_j, not all literals l_j^1, l_j^2, l_j^3 are evaluated to false, and hence iff there is an explanation that embeds a causal ordering from a_0 to every a_{c_j}. □

Theorem 1 shows that the complexity of diagnosis increases with the size of the observation, and Theorem 2 shows that even a simpler problem such as existence of an explanation can rapidly become difficult to solve. To handle this complexity, we can reduce the size of the observed alphabet, which will hopefully produce smaller observations, or try to split a problem into smaller ones and then combine the results. The following proposition shows that limiting the observation capacities of the system does not produce wrong negatives for the existence problem.

Proposition 5 *[10] Let H be a HMSC, Σ_{obs} be an observation alphabet, and O be an observation. Let $\Sigma'_{obs} \subseteq \Sigma_{obs}$. Then if $\Pi_{\Sigma'_{obs}}(O)$ has no explanation from H w.r.t. Σ'_{obs}, then O has no explanation w.r.t. Σ_{obs} from H.*

Proof: Suppose that there exists no explanation for an observation O, with alphabet Σ'_{obs}, but that we can build an automaton $\mathcal{A}_{O,H}$ with observation alphabet Σ_{obs}. In particular, it means that for every path ρ of H, there is no embedding of O into M_ρ w.r.t. Σ'_{obs}. If no embedding exists, then either there exists a process p such that $\Pi_{\Sigma'_{obs} \cap \Sigma_p}(O)$ is not a prefix of $\Pi_{\Sigma'_{obs} \cap \Sigma_p}(M)$, and then M does not embed O with observation alphabet Σ_{obs}, or there exist two events $x \leq_O y$ with labels $\alpha(x), \alpha(y) \in \Sigma'_{obs}$. As projection preserves ordering, x and y are ordered both in O and $\Pi_{\Sigma'_{obs}}(O)$, and M can not embed O. □

This property possibly reduces the time needed to give a negative answer to the existence problem and will be useful later on in the paper to divide the diagnosis problem into smaller tasks, and hence reduce the time needed to build a complete diagnosis.

4.2 Splitting the diagnosis problem

So far, the diagnosis framework proposed is centralized (all observations are sent to a central diagnoser that computes the generator for the set of explanations) and offline (the production of a diagnosis is performed once for all from a fixed observation). The main objective of the approach is to perform all calculi on partial order models, and avoid the state space explosion due to an interleaved search in the execution model. From Theorem 1, we can see that the interleaved behaviors of the model are never studied, but that in worst cases, one may have to consider all linearizations of an observation. Furthermore, the centralized diagnosis amounts to build an automaton of exponential size in the number of considered processes.

A solution to resolve this problem is to split the diagnosis computation into several simpler sub-problems that can be easily distributed to solve the diagnosis concurrently. In this section, we will show a distribution schema that does not change the result of centralized diagnosis, but can allow for a faster detection of non-existence when no explanation of an observation exists.

The main idea is to separate the diagnosis into smaller problems using projections of the observation on subset of processes. From proposition 5, we know that it is sufficient to find no diagnosis for an observation O projected on an alphabet $\Sigma' \subseteq \Sigma_{obs}$ to be sure that no diagnosis exists for O. In particular, this applies to the case when Σ is the restriction of Σ'_{obs} to events that are observable on a chosen subset of processes. The hard technical point is then to combine local diagnosis, and show that their combination produces the same result as the centralized version.

Let $p, q \in \mathcal{P}$ be a pair of instances and $O = (O, \leq_O, \alpha_0, \mu_O)$ be an observation. The local diagnosis for instances p, q is the automaton $\mathcal{A}_{p,q} = \mathcal{A}_{\pi_{\Sigma'}(O),H}$ with the observation alphabet $\Sigma' = \Sigma_{obs} \cap (\Sigma_p \cup \Sigma_q)$. Since an explanation of an observation for some alphabet Σ is still an explanation for any alphabet $\Sigma' \subseteq \Sigma$, we have that $\mathbb{L}_{H,\mathcal{A}_{O,H}} \subseteq \mathbb{L}_{H,\mathcal{A}_{p,q}}$. Hence, a finer diagnosis can be obtained from successive compositions of local diagnosis. This composition $\otimes$ is simply an intersection, defined as a synchronous product of two diagnosis automata.

Definition 11 *Let $\mathcal{A}_{O,H} = (Q, \delta, \mathcal{M}, q_0, F)$ and $\mathcal{A}'_{O',H} = (Q', \delta', \mathcal{M}, q'_0, F')$ be two diagnosis automata. The* synchronous product *of $\mathcal{A}_{O,H}$ and $\mathcal{A}'_{O',H}$ is denoted by $\mathcal{A}_{O,H} \otimes \mathcal{A}'_{O',H}$, and is the automaton $\mathcal{A}_{O,H} \otimes \mathcal{A}'_{O',H} = (Q \times Q', \delta'', \mathcal{M}, (q_0, q'_0), F \times F')$, where $\big((v, w), M, (v', w')\big)$ is a transition of δ'' iff $(v, M, v') \in \delta$ and $(w, M, w') \in \delta'$.*

The next proposition shows that when a run belongs to every $\mathcal{A}_{p,q}$, for pairs of processes in $\mathcal{P}_{obs}$ then it is an explanation of O:

Theorem 3 *For every HMSC H and observation O, we have*
$\mathbb{L}_{H,\mathcal{A}_{O,H}} = \mathbb{L}_{H,\mathcal{A}_{\bigotimes}}$*, where* $\mathcal{A}_{\bigotimes} = \bigotimes_{p \neq q \in \mathcal{P}_{Obs}} \mathcal{A}_{p,q}$.

Proof : For every pair of processes p, q in $\mathcal{P}_{obs}$, the observation alphabet $\Sigma' = \Sigma_{obs} \cap (\Sigma_p \cup \Sigma_q)$ is contained in Σ_{obs}. Hence, from corollary 1, it is obvious that for every path ρ of H, if M_ρ is an explanation of an observation O, then M_ρ is also an explanation for the projection $\Pi'_{\Sigma}(O)$ on a the smaller observation alphabet Σ'. Hence $\mathbb{P}_{O,H} \subseteq \mathbb{L}_{H,\mathcal{A}_{p,q}}$ for all $p, q \in \mathcal{P}_{Obs}$, and we have the first inclusion $\mathbb{L}_{H,\mathcal{A}_{O,H}} = \mathbb{P}_{O,H} \subseteq \mathbb{L}_{H,\mathcal{A}_{\bigotimes}}$.

For the second inclusion, let $\rho \in \mathbb{L}_{H,\mathcal{A}_{\bigotimes}}$. This means in particular that $\rho \in \mathbb{L}_{H,\mathcal{A}_{p,q}}$ for every pair p, q in $\mathcal{P}_{obs}$. Let $h_{O,M_\rho} : E_O \to E_{M_\rho}$ be the (unique) function such that the k-th event of E_0 on instance p is sent by h_{O,M_ρ} to the k-th event of $\alpha^{-1}_{M_\rho}(\Sigma_{Obs})$ on instance p for all k and $p \in \mathcal{P}_{Obs}$. We can denote by $h_{p,q}$ the restriction of h_{O,M_ρ} to events located on $p, q \in \mathcal{P}_{Obs}$. Clearly, $h_{p,q}$ is also the unique mapping defined by $\pi_{\Sigma_{Obs} \cap (\Sigma_p \cup \Sigma_q)}(O) \triangleright M_\rho$.

We have easily that for every $p, q \in \mathcal{P}_{Obs}$, for every event e located on p, $\alpha_O(e) = \alpha_{M_\rho}(h_{O,M_\rho}(e)) = \alpha_{M_\rho}(h_{p,q}(e))$. Then, for every pair of events $e, f \in E_O$ located on p, we have that $e \leq_O f$ implies that $h_{O,M_\rho}(e) = h_{p,q}(e) \leq_{M_\rho} h_{p,q}(f) = h_{O,M_\rho}(f)$.

Now, assume that $e, f \in O$ are causally ordered and located on different instances, p and q. Since $\rho \in \mathbb{L}_{H,\mathcal{A}_{p,q}}$, and since the projection of O on a pair of processes preserves the ordering on projected events, we have $h_{O,M_\rho}(e) = h_{p,q}(e) \leq h_{p,q}(f) = h_{O,M_\rho}(f)$.

At last, assume by contradiction that $h_{O,M_\rho}(E_O)$ is not a prefix of $\pi_{\Sigma_{obs}}(M_\rho)$. Then there exists $e, f \in M_\rho$, located on processes p and q, and of type in Σ_{obs} such that $e \leq_M f$, $e \notin h_{O,M_\rho}(E_O)$ and $f \in h_{O,M_\rho}(E_O)$. However, since $\rho \in \mathbb{L}_{H,\mathcal{A}_{p,q}}$, we know that $h_{p,q}(E_O)$ is a prefix of some sub-order of $\pi_{p,q}(\pi_{\Sigma_{obs}}(M_\rho))$, which gives us a contradiction. □

From Theorem 1, we know that the size of $\mathcal{A}_{p,q}$ is in $O(|O|^{2|\mathcal{P}|}.|H|)$. Let us detail how this can impact the diagnosis and existence problems. An immediate idea stemming from theorem 3 is to split computation of $\mathcal{A}_{O,H}$ from O and H into $|\mathcal{P}_{Obs}|^2$ sub-problems, that is compute $\mathcal{A}_{p,q}$ from $\Pi_{\Sigma_{obs}\cap(\Sigma_p\cup\Sigma_q)}(O)$ for each pair of processes $p \neq q \in \mathcal{P}_{Obs}$, and then compute the product of these local diagnosis. To obtain the final diagnosis, we then have to compute a product of $|\mathcal{P}_{Obs}|^2$ automata if none of the local results is empty. Note that the size of a local automaton is not necessarily smaller than the size the original diagnosis automaton computed in the centralized version. The size of the whole product is exactly the same as in the original version. However, the time needed to complete diagnosis is enhanced if some local problems can be computed in parallel, or in any case when one of the local diagnosis returns an empty diagnosis. Indeed, if $\mathbb{L}_{H,\mathcal{A}_{p,q}} = \emptyset$ for some pair $p, q \in \mathcal{P}^2_{Obs}$, then $\mathbb{L}_{H,\mathcal{A}_{O,H}} = \emptyset$, and it is useless continuing the computation of all other local diagnosis automata. □

This allows for a decentralized version of our initial diagnosis architecture, depicted in Figure 11.

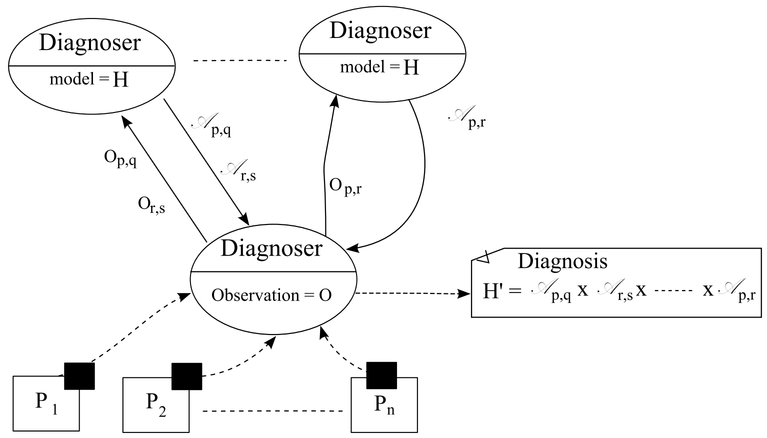


Fig. 11 Decentralized Diagnosis/Existence problem

There is still a central diagnoser that collects all the observation. Its role is to compute a projection $\Pi_{\Sigma_{obs}\cap(\Sigma_p\cup\Sigma_q)}(O)$, and send it to local diagnosers. Each local diagnoser possess a copy of H, and upon reception of an observation $\Pi_{\Sigma_{obs}\cap(\Sigma_p\cup\Sigma_q)}(O)$, computes a diagnosis $\mathcal{A}_{p,q}$ and returns it to the central diagnoser. The central diagnoser computes incrementally the product $\mathcal{A}_{\bigotimes}$ of the $|\mathcal{P}_{Obs}|^2$ after each reception of a local result, and returns the final result when all local diagnosis have been completed. It can also return the empty diagnosis as soon as one local diagnoser returns an empty diagnosis. Note that there is no need for $|\mathcal{P}_{Obs}|^2$ local diagnosers, as each of them can compute more that one local diagnosis.

We have shown in theorem 3 that the diagnosis problem can be brought back to a product of smaller local diagnosis problems. A question that immediately arises is whether the existence problem can also be decentralized, and even more interesting, be seen as the conjunction of Boolean answers to local existence problems. For the existence problem, we know that as soon as a local diagnosis is empty, H provides no explanation for O. However, a product of *all* local results *have to be computed* to make sure that a global explanation exists for O in H. As for diagnosis, the product can be built incrementally from local diagnosis computed concurrently, and a negative answer can be returned as soon as a local diagnoser returns an empty diagnosis. Note however that there are cases where all diagnosis compute a diagnosis automaton with non-empty language (i.e. $\mathbb{L}_{H,\mathcal{A}_{p,q}} \neq \emptyset$ for every pair $p, q \in \mathcal{P}^2_{Obs}$) but where the global diagnosis is nevertheless an automaton with empty language (i.e. $\mathbb{L}_{H,\mathcal{A}_{O,H}} = \bigcap_{p,q \in \mathcal{P}^2_{Obs}} \mathbb{L}_{H,\mathcal{A}_{p,q}} = \emptyset)$.

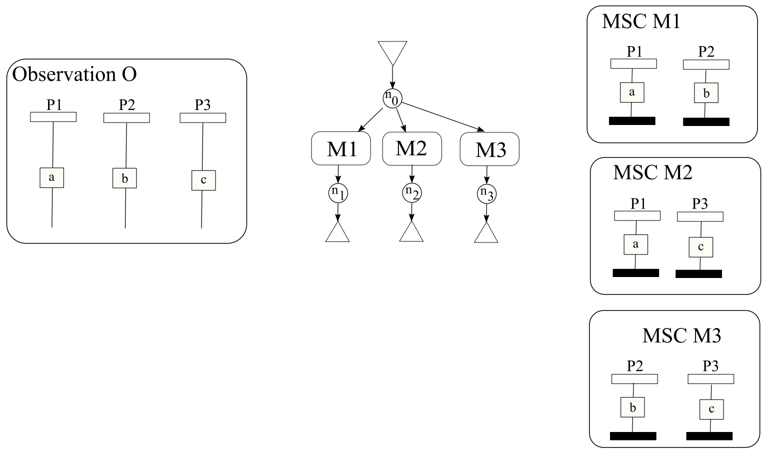


Fig. 12 Example showing that computing a product is needed for decentralized existence

Example 8 Consider for instance the HMSC H and the observation O of figure 12. For each pair of processes p, q in $\{P1, P2, P3\} \times \{P1, P2, P3\}$, it is possible to find an explanation for $\Pi_{\Sigma_{obs} \cap (\Sigma_p \cup \Sigma_q)}(O)$. However, H does not contain execution that exhibits at the same time events a, b, c.

Another solution to decentralized diagnosis is to consider process by process diagnosis, that is compute the automaton $\mathcal{A}_p = \mathcal{A}_{\pi_{\Sigma_{obs} \cap \Sigma_p}(O),H}$ that provides all explanations of H for $\pi_{\Sigma_{obs} \cap \Sigma_p}(O)$ for each $p \in \mathcal{P}_{obs}$. Computing this set of automata gives a less precise solution than with pairs of processes, because ordering between events of O located on distinct processes cannot be used to discriminate some paths. Indeed, in general $\mathbb{L}_{\mathcal{A}_{p,q}} \subseteq \mathbb{L}_{\mathcal{A}_p} \otimes \mathbb{L}_{\mathcal{A}_q}$, but equality does not hold. However, the initial step computing $\mathcal{A}_p$ is performed with complexity in $O(|O| \times |H|)$. Notice that $\mathcal{A}_{p,q}$ has to be computed only for those $p \neq q \in \mathcal{P}_{obs}$ for which there exist two events $e, f \in E_O$ respectively located on p and q and such that $e \leq_O f$. If no such ordering from p to q exists in O, then $\mathcal{A}_{p,q} = \mathcal{A}_p \otimes \mathcal{A}_q$. This leaves room for improving efficiency of decentralized diagnosis.

5 Online diagnosis

So far, we have only considered offline diagnosis and offline existence problems, that is finding a posteriori from an observation what occurred during the use of a system, or if a model provides any explanation at all. We now address these two problems online, that is compute solutions incrementally as observed events arrive from probes. A naive approach to solve these two problems online with the framework described in section 4 would be to remember the whole observation, and to apply the offline diagnosis algorithm with the whole observation every time a new event is observed. This technique is clearly inefficient, but proves the existence of an online diagnosis algorithm. The cost of such solution is:

$$C = |H|. \sum_{i \in 1..|O|} i^{|\mathcal{P}| \times |\mathcal{P}_{Obs}|}$$

This solution is of course too costly, and calls for an incremental and efficient search for explanations. The main objective is then to reuse the part of the diagnosis performed at step n during step $n+1$. In this section, we explore the possible solutions to perform efficiently online diagnosis and existence checking, and the memory needed to run such algorithms. As offline and online diagnosis algorithms should return the same results, we cannot expect the online solution to be faster than the offline one, nor to return smaller results. However, in an online setting, it is worth noticing that some parts of the diagnosis that have been computed and can not be refuted by future observations can be stored on a disk, while the rest of the diagnosis is kept in memory for future analysis. For the online existence problem, the non-refutable part of the diagnosis can simply be forgotten. The first thing to define in an online framework is the information that must be kept in memory. We will show that at each step of online diagnosis, only a part of the diagnosis automaton built so far need to be remembered to continue a diagnosis on the fly. We address the two following problems; The first one is the online construction of a diagnosis, that is the incremental construction of an object similar to the output of offline diagnosers. The second question addressed is the online detection of existence of an explanation. Obviously, the second problem is a specialization of the first one, and needs recording less information than for online construction.

Before entering into the technical details of diagnosis, we should first describe how new occurrences of observed events are collected and assembled to produce a coherent observation. Let us consider again the picture 1 in section 1. A diagnoser collects all information coming from observed processes.

In offline diagnosis, we consider that diagnosis starts from an already built observation. In the online setting, the diagnoser repeatedly receive new observed events from probes, and updates observation and diagnosis accordingly. Now, the only assumption that we have on communications from probes to the diagnoser is that the communication channels are FIFO and asynchronous. Hence, two events observed on the same machine are necessarily ordered. Now, if two events e, f have been observed on distinct processes and e precedes f in the execution, it might be the case that f is received by the observed **before** e. As already mentioned, observed events can be tagged by information such as vectorial clock to record a part of the ordering among them, or at minimal a sequence number on the executing process. Let us call O the observation that have been built so far. When an event f is received, if its tag indicates that an observable event e that does not belong to O precedes it, then f is not appended to the observation, and the diagnoser waits for the arrival of e to append f. If the tags associated to f are consistent with the events observed so far in O, and in particular it does not mention the existence of an unreceived observable event in the past of f, then f can be appended to O without waiting. Hence, a diagnoser may have to memorize some events before using them for diagnosis. In the rest of the paper, we will consider that when an event is appended to an observation O, then all its predecessors in O are known, and have also been appended. In the rest of the paper, we will denote by $O.e$ the observation obtained by appending an event e to O, and will suppose that all predecessors of e already appear in O.

5.1 Safe parts of a diagnosis

In this section, we first define formally the part of a diagnosis that have no influence on future steps of the online construction of a diagnosis. These parts are called the *safe parts* of a diagnosis.

Definition 12 *Let $\mathcal{A}_{O,H} = (Q, \delta, \mathcal{M}, q_0, F)$ be the automaton computed for the diagnosis of observation O from HMSC H. Let F be a set of transitions of $\mathcal{A}_{O,H}$. We say that F is a* safe part *of $\mathcal{A}_{O,H}$ if and only if* every observable event of F has been observed: *for every transition $q_{k-1} \xrightarrow{M_k} q_k \in F$, observable event $e \in M_k$, and every path $\rho = q_0 \xrightarrow{M_1} q_1 \dots q_{k-1} \xrightarrow{M_k} q_k$ of $\mathcal{A}_{O,H}$, there exists f in O such that $e = h_{O,M_\rho}(f)$ (where h_{O,M_ρ} is the unique embedding of O in M_ρ).*

Notice that even if safe parts are defined in terms of paths, two paths ρ_1, ρ_2 going through the same transition $(n, E, g) \xrightarrow{M} (n, E', g') \in F$ have the same impact on F being safe or not. Indeed, for $i \in \{1, 2\}$ and an observable event $e \in M$, there exists f in O such that $e = h_{O,M_{\rho_i}}(f)$ if and only if $e \in E' \setminus E$. But this last expression, does not depend on i, hence $e = h_{O,M_{\rho_1}}(f)$ if and only if $e = h_{O,M_{\rho_2}}(f)$.

It implies that the union of two safe parts is a safe part, and hence there exists a *maximal safe part* $F_{\max}$. For representation reason, we will be interested only in *maximal connected safe parts*, that is the connected components of $F_{\max}$. We will say that two transitions $q_1 \xrightarrow{M_1} q_1'$ and $q_2 \xrightarrow{M_2} q_2'$ are are *connected* if $q_1 = q_2'$ or $q_2 = q_1'$.

It is easy to build $F_{\max}$ in linear time in the number of transitions: $(n, E, g) \xrightarrow{M} (n, E', g') \in F_{\max}$ iff $|E'| - |E| = |\pi_{\Sigma_O}(M)|$ (there are as many events in $E' \setminus E$ as observable events in M). One can also compute every connected component of $F_{\max}$ in linear time in $F_{\max}$. In the rest of the paper, we will say that an event e in a MSC M_ρ is *safe* if it has been observed (i.e. there exists an embedding h_{O,M_ρ} and an event $f \in E_O$ such that $h_{O,M_\rho}(f) = e$).

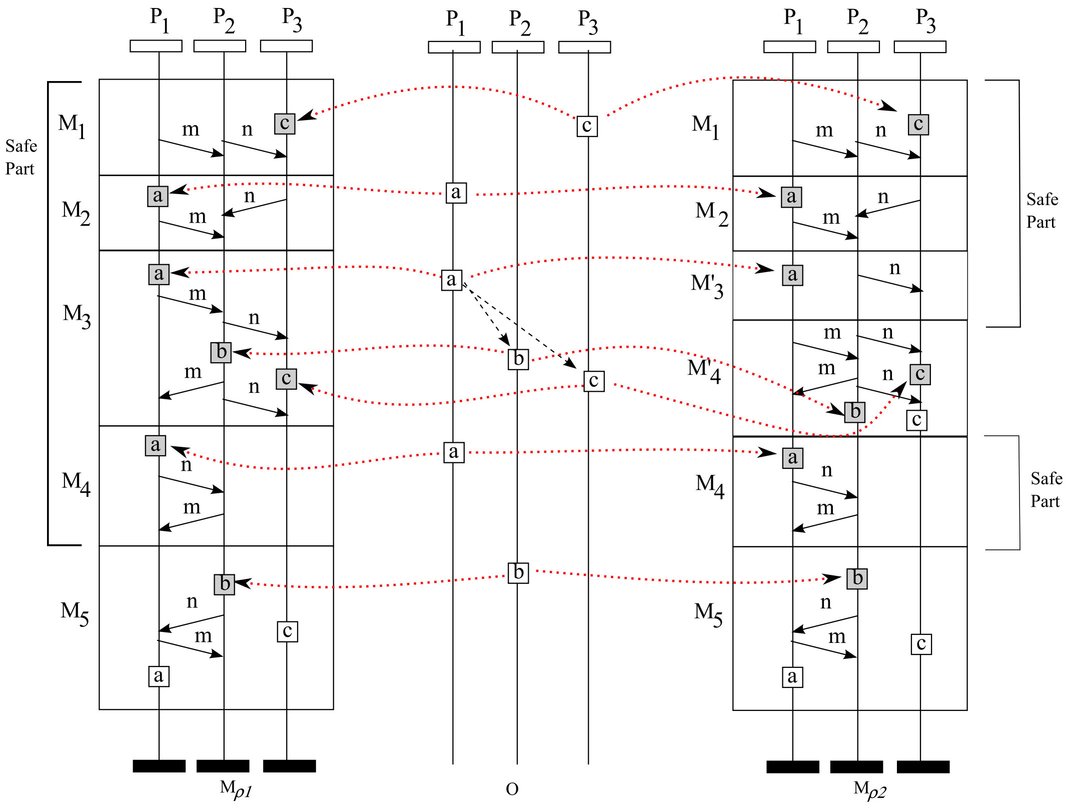


Fig. 13 Observations and safe parts

Example 9 Figure 13 shows an example of safe parts for two paths of the same HMSC, $\rho_1 = n_0 \xrightarrow{M_1} n_1 \xrightarrow{M_2} n_2 \xrightarrow{M_3} n_3 \xrightarrow{M_4} n_4 \xrightarrow{M_5} n_5$ *and* $\rho_2 = n_0 \xrightarrow{M_1} n_1 \xrightarrow{M_2} n_2 \xrightarrow{M_3'} n_3' \xrightarrow{M_4'} n_3 \xrightarrow{M_4} n_4 \xrightarrow{M_5} n_5$*. The observation alphabet is* $\Sigma_{obs} = \{P_1(a), P_2(b), P_3(c)\}$*.* ρ_1 *and* ρ_2 *are valid explanations for the observation O depicted in the center of the figure, as there exists embeddings from O to* M_{ρ_1} *and* M_{ρ_2}*, depicted by the dotted arrows from events of O to their image in the corresponding MSC. Safe events are denoted by grey squares. The maximal safe part of* ρ_1 *is* $\{n_0 \xrightarrow{M_1} n_1 \xrightarrow{M_2} n_2 \xrightarrow{M_3} n_3 \xrightarrow{M_4} n_4\}$*. The maximal safe part of* ρ_2 *is* $\{n_0 \xrightarrow{M_1} n_1 \xrightarrow{M_2} n_2 \xrightarrow{M_3'} n_3'\} \cup \{n_3 \xrightarrow{M_4} n_4\}$*. Note that the maximal* *safe part in ρ_2 is not made of consecutive transitions, as M_1, M_2, M_3', M_4 have been completely observed, but not M_4'.*

Note also from this figure that even though in the explanation $M_1 \circ M_2 \circ M_3 \circ M_4 \circ M_5$ some events are ordered, as for instance atomic actions $P_2(b)$ in M_3 and $P_1(a)$ in M_4, this ordering does not appear in the observation. Last, note that the last occurrences of $P_1(a)$ and $P_1(c)$ in MSC M_5 are not mapped to any event of O in path $\rho 1$, and similarly for the last occurrence of $P_1(a)$ and the last two occurrences of $P_1(c)$ in path ρ_2. Even though, M_{ρ_1} and M_{ρ_2} are explanations of O, as these actions can be considered as not yet observed.

Proposition 6 *Let $O = (E_O, \leq_O, \alpha_O, \mu_O, \phi_O)$ be an observation, and let $\rho = t_1.t_2 \ldots t_n$ be a path of $\mathbb{L}_{\mathcal{A}_{O,H}}$. Then for every O' such that O is a prefix of O', and $E_{O'} \setminus E_O = \{e\}$, there exists an embedding of O' in M_ρ if and only if there exists a transition $t_k = (n, E, g) \xrightarrow{M} (n', E', g')$ of ρ such that:*

- *The minimal event f on $\phi(e)$ in $\pi_{\Sigma_O}(M) \setminus h_{O,M_\rho}(E' \setminus E)$ is such that $\alpha(f) = \alpha(e)$. Intuitively, f is the first unobserved event in M on process $\phi(e)$.*
- *For each $e' <_{O'} e$, there exists $f' \in M$, such that $f' < f$, and either $e' \in g(\phi(f'))$, or $h_{O,M_\rho}(f') = e'$*
- *For every transition $t_i = (n_i, E_i, g_i) \xrightarrow{M_i} (n_i', E_i', g_i')$ in $t_1 \ldots t_{k-1}$, $\pi_{\Sigma_O,\phi(e)}(M) = \pi_{\Sigma_O,\phi(e)}(E' \setminus E)$, where $\pi_{\Sigma_O,p}(X)$ denotes the restriction of X to events located on process p and with labels in Σ_O. Intuitively, this last property means that M is the first MSC in path ρ that contains an unsafe event on process $\phi(e)$.*

. Notice that since $|E'| - |E| \neq |\pi_O(M)|$, $t \notin F_{\max}$.

This property is important for online diagnosis. It means that when diagnosis is built incrementally, $F_{\max}$ does not influence the next steps of the construction (this holds even for connected component of $F_{\max}$ that do not contain the initial state). Hence, we can just forget (or store somewhere for later use) $F_{\max}$, as it will not influence the construction of the diagnosis, and remember the connections among unsafe transitions provided by the safe parts. Checking that an event does not have a predecessor on its process can be done efficiently by searching backward unobserved events in the diagnosis automaton, and verifying that all predecessors of an event appear along a path can be checked locally to a transition using function g. Hence, the search for an appropriate embedding can be performed locally to each unsafe transition.

So far the properties addressed show how to verify that an increment to an existing observation is still embedded in a path of a diagnosis automaton. Now, as new events arrive, the observation and the diagnosis automaton should be updated jointly. We first show how to update and extend an existing path, and then adapt this extension mechanism to the whole diagnosis automata.

Proposition 7 *Let $\rho = t_1 \ldots t_k$ be a path of $\mathbb{L}_{\mathcal{A}_{O,H}}$ such that there exists an embedding of $O' = O.e$ in M_ρ, and e is mapped to an event f of transition*

$t_i = (n, E, g) \xrightarrow{M_i} (n_i, E_i, g_i), i \in 1..k$. Let $\rho' = t_1 \dots t_{i-1}.t'_i.t'_{i+1} \dots t'_k$ where $t'_i = (n, E, g) \xrightarrow{M_i} (n_i, E_i \cup \{e\}, g'_i)$, and every t'_j is obtained by adding $\{e\}$ to the observed event set, and updating function g as in definition 10. Then ρ' is a path of $\mathbb{L}_{\mathcal{A}_{O',H}}$.

This proposition shows how to update a path (and hence a diagnosis automaton) when an embedding of $O.e$ exists in this path. More precisely, when a function g_j is updated to g'_j with $j \geq i$, we will have $e \in g'(p)$ iff there exists an event f located on process p and such that $e \leq f$ in $M_i \circ \dots M_j$. However, in some cases, all events along a path ρ may have been observed. Hence, there is no unsafe event matching the freshly observed event e in M_ρ, and one have to append new transitions to this path to provide explanations for $O.e$.

Definition 13 *Let ρ be a path of $\mathbb{L}_{\mathcal{A}_{O,H}}$ ending in state (n, E, g), and let $\gamma = t_1 \dots . t_k$ be a path of H starting from node n where each $t_i, i \in 1..k$ is of the form $n_{i-1} \xrightarrow{M_i} n_i$. The* extension *of ρ with γ is a path $\rho.t'_1 \dots t'_k$, where t'_1 is of the form $(n, E, g) \xrightarrow{M_1} (n_1, E, g_1)$, and each $t'_i, i \in 2..k$ is of the form $(n_{i-1}, E, g_{i-1}) \xrightarrow{M_i} (n_i, E, g_i)$, where for every $p \in \mathcal{P}$, $e \in E$ and $j \in 1..k$, we have $e \in g_j(p)$ iff there exists a process q and two events f, f' in $M_1 \circ \dots \circ M_j$ such that $e \in g(q)$, $\phi(f) = q$, $\phi(f') = p$ and $f \leq f'$.*

As shown in the definition, the extension mechanism only updates function g, as it involves no new observation. It is straightforward to see that if ρ is a path of $\mathbb{L}_{\mathcal{A}_{O,H}}$, then any extension of ρ is also a path of $\mathbb{L}_{\mathcal{A}_{O,H}}$.

Proposition 8 *Let ρ be a path of $\mathbb{L}_{\mathcal{A}_{O,H}}$ ending in state $s = (n, E, g)$ such that all events of process $\phi(e)$ in M_ρ have been observed in O. Let $O' = O.e$, and ρ' be the extension of ρ with a path γ of H. Then, there is an embedding of O' in $M_{\rho'}$ iff:*

- *there exists an event f in M_γ such that $\alpha(f) = \alpha(e)$ and f is the first observable event on process $\phi(e)$ in M_γ.*
- *For each $e' <_{O'} e$, there exists $f' \in M_\gamma$, $f' < f$, and $e' \in g(\phi(f'))$.*

Proposition 8 shows how to test if a candidate path can be used to explain a new observation. Note however that the set of paths containing an adequate observable event with correct ordering can be infinite. Hence, to perform online diagnosis, we next show that it is sufficient to work with a reduced version of the diagnosis automaton, containing minimal acyclic paths.

5.2 Summarizing a diagnosis

As already mentioned in section 4, the diagnosis automaton built from a HMSC and from an observation does not provide the smallest paths leading to an explanation. First, the construction of $\mathcal{A}_{O,H}$ may also produce states that are not co-accessible, i.e. from which a final state where the whole observation O has been explained can never be reached. These states can however easily be detected and removed. On the other hand, $\mathcal{A}_{O,H}$ may contain loops. In particular, any transition from a state in which the observation have been completely explained (states of the form (n, E, g) with $E = E_O$) is allowed. These transitions make sense in an offline context where we are interested by a generator of all explanations of an observation contained in a model. However, in an online context, the observation available at a given moment is only a prefix of an eventual observation. We will show in the sequel that it is sufficient to remember acyclic paths to build a complete diagnosis. For this reason, we will mainly focus on minimal and acyclic paths containing an observation.

Definition 14 *Let $\rho = (n_0, E_0, g_0) \xrightarrow{M_0} (n_1, E_1, g_1) \xrightarrow{M_1} \dots \xrightarrow{M_{k-1}} (n_k, E_k, g_k)$ be a path of a diagnosis automaton such that $O \triangleright_{\Sigma_{obs}} M_\rho$. ρ is said to be a* minimal acyclic explanation *of O if and only if:*

- *there is no prefix of ρ that explains O,*
- *ρ does not contain subsequences of the form $(n_i, E, g) \xrightarrow{M_i} (n_{i+1}, E, g) \dots \xrightarrow{M_j} (n_i, E, g)$.*

In particular, this definition means that a minimal acyclic path can not contain a cycle of the original HMSC H in which the set of observed events or the set of observed causalities do not progress. It can not either contain suffixes of the form $(n_i, E, g_i) \xrightarrow{M_i} (n_{i+1}, E, g_{i+1}) \dots \xrightarrow{M_{j-1}} (n_j, E, g_j)$, as the observation is already explained by smaller paths.

Example 10 Consider for instance the diagnosis automaton built in Figure 9. The path $q_0 \xrightarrow{M1} q_1 \xrightarrow{M3} q_2 \xrightarrow{M3} q_3 \xrightarrow{M2} q_4$ is a minimal acyclic path, but $q_0 \xrightarrow{M1} q_1 \xrightarrow{M3} q_2 \xrightarrow{M3} q_3 \xrightarrow{M3} q_3 \xrightarrow{M2} q_4$ is not.

Proposition 8 shows how to extend a path ρ with a sequence of transitions ρ' of H of arbitrary size. However, we will see in the sequel that it is sufficient to work with summaries, that is abstractions of acyclic and minimal paths to find a correct diagnosis. To find explanations in an extension of a path ρ, we hence simply have to consider the minimal and acyclic path of H containing an explanation for the searched event e.

Proposition 9 *Let ρ be a minimal acyclic path of $\mathcal{A}_{O,H}$ (i.e. embeds O) such that all events of process $\phi(e)$ in M_ρ have been observed in O. Let $O' = O.e$. Then, finding the set P_e of all paths γ of H such that extensions of ρ with γ are minimal and acyclic and embed O' can be done in $O(|\mathcal{P}|.|H|.2^{|\mathcal{P}|})$.*

Proof: We have to search an event f such that $\alpha(f) = \alpha(e)$ in all acyclic path that start from the final node reached by ρ. We furthermore have to verify that f is the first observable event on $\phi(e)$, and that the image of all predecessors of e via h_{O,M_ρ} will be predecessors of f. For this, we can build an automaton that remembers the current state of the HMSC visited, and for each predecessor e' of e and for each process $p \in \mathcal{P}|$ whether e' has a successor on p or not. The construction stops when:

- an event which is not labeled as e is found on process $\phi(e)$,
- the next transition to play brings back to an already visited state (hence we are going to build a cycle, and the followed path is not minimal),
- an event that is labeled as e and is minimal on process $\phi(e)$ is found, but with incorrect set of predecessors.
- an event that is labeled as e and is minimal on process $\phi(e)$ is found, with correct set of predecessors (this path is successful, minimal,; and acyclic).

Hence, there are at most $|\mathcal{P}|.|H|.2^{|\mathcal{P}|}$ such states to explore to find minimal extensions explaining $O.e$. $\square$

This construction guarantees that if a path ρ is minimal and acyclic, then for any $\rho' \in P_e$, we have that $\rho.\rho'$ is also minimal and acyclic. Note that the construction of the proof is not a set of paths, but an automaton. We will use this automaton in the sequel. We will then write $Ext(s, e)$ to refer to the automaton built from a state s to find a minimal and acyclic explanation containing a new event e.

The next definition and propositions show that the diagnosis automaton built in offline diagnosis in section 4 is an acyclic graph decorated by unobservable loops. Hence, it is possible to work on a reduced form of diagnosis that forgets these loops, and to get back to a complete diagnosis after the construction of the acyclic version.

Definition 15 *Let $\mathcal{A}_{O,H} = (Q, \delta, \mathcal{M}, q_0, F)$ be the automaton computed for the diagnosis of observation O from HMSC H. A* silent transition *of $\mathcal{A}_{O,H}$ is a transition labeled by a MSC that does not contain observable events, and does not change the value of $g(p)$ for every $p \in \mathcal{P}$. The* summary *of $\mathcal{A}_{O,H}$ is the automaton $\overline{\mathcal{A}_{O,H}}$ obtained by:*

1) removing all cycles over silent transitions,
2) removing transitions from accepting states,
3) restricting the obtained automaton to co-accessible states.

Note that the first step of summary construction does not only consist in removing silent transitions. Some silent transitions modify functions $g(p)$ for some $p \in \mathcal{P}$, and are needed to ensure a correct ordering among observed events. Hence, only silent cycles, which obviously do not change the set of observed events nor function g should be removed. Computing $\overline{\mathcal{A}_{O,H}}$ then consists in a traversal of $\mathcal{A}_{O,H}$ that disallows transitions creating a silent cycle and transitions from accepting states. The obtained tree is then reduced to co-accessible states. Computing a summary can hence be done in linear time in the number of transitions of $\mathcal{A}_{O,H}$.

Example 11 Consider the example of Figure 9. Obtaining a summary from this automaton simply consists in removing the self loops labeled by $M3$ on states q_0 and q_3.

Proposition 10 *Let $\mathcal{A}_{O,H}$ be a diagnosis automaton restricted to co-accessible states. Then $\overline{\mathcal{A}_{O,H}}$ is unique and acyclic. Furthermore, computing $\mathcal{A}_{O,H}$ from $\overline{\mathcal{A}_{O,H}}$ and vice versa can be done in linear time.*

Proof: As all silent cycles have been removed, each path of $\overline{\mathcal{A}_{O,H}}$ has a bounded number of silent transitions, and a bounded number of non-silent transitions (at most $|O|$). Each transition in $\overline{\mathcal{A}_{O,H}}$ is a move towards a non-silent transition, and when a non silent transition is taken, or when a transition modifies function g, it is impossible to get back to previously traversed states. As progress (growth of the number of mapped events in O) is guaranteed after at most $|H|$ transitions, and as all transitions are disallowed when all events of O are mapped, then $\overline{\mathcal{A}_{O,H}}$ is finite and acyclic.

Note that in the diagnosis automata, cycles are necessarily composed of silent transitions. Hence, the computation of a summary does not affect reachability of observable transitions (except in accepting states, from which summaries forbid all transitions). The summary can then be seen as the result of a tau* minimization (a minimization procedure for transition systems that considers weak bisimulation as equivalence relation on states [5]), followed by addition of acyclic unobservable paths needed to reach all observable transitions. This procedure guarantees uniqueness of the obtained summary.

The transformation of diagnosis into acyclic automaton is functional (it is simply a projection of $\mathcal{A}_{O,H}$ on a subset of its transitions). To get back to the initial diagnosis, it suffices to add all transitions from accepting states to the acyclic diagnosis, and to connect to each state of $\overline{\mathcal{A}_{O,H}}$ the silent strongly connected components of H. For a fixed H and O, this construction is also a deterministic function, that adds at most $|H|$ transitions per state of $\overline{\mathcal{A}_{O,H}}$.

□

This proposition is important, and shows that we can perform online diagnosis by computing incrementally an acyclic summary, and then obtain a complete diagnosis automaton. Of course, the complete diagnosis does not have to be computed at each observation, and should be returned on demand to users of the diagnosis framework. Note also that a summary can be much smaller than the original automaton which can be of size up to $(|H|.|\overline{\mathcal{A}_{O,H}}|)$.

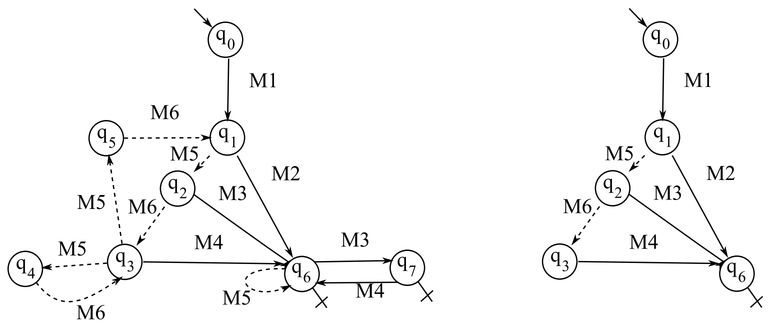


Fig. 14 A diagnosis automaton and its summary

Example 12 Consider the diagnosis automaton in the left part of Figure 14. Initial state is symbolized by a circle with an incoming arrow, final states by circles with a cross. Transitions from one state to another are symbolized by arrows, and silent transitions by dashed arrows. Note on this example that all *successors of an accepting state are also accepting. Note also that cycles are exclusively composed of silent transitions. The automaton at the right of the figure is the summary of the automaton at the left of the figure. It is obtained by removing transitions* (q_4, M_6, q_3), (q_5, M_6, q_1) *and all transitions leaving* q_6, *and then restricting the obtained automaton to coaccessible states. Note that transitions* (q_1, M_5, q_2) *and* (q_2, M_6, q_3) *are silent, but remain in the summary, as they are needed to reach observable transitions labeled by* $M2, M3, M4$.

5.3 Abstraction of a diagnosis

So far, we have seen that it is sufficient to build a summary of a diagnoser to be able to build the complete automaton that gives all explanations of an observation contained in an HMSC. We have also seen that when a new event is observed, some transitions of a diagnosis automaton, called the safe part, do not influence the existence of an embedding. Hence, these safe parts can simply be forgotten if the goal is to check existence of an explanation or stored on disk for later use if we want to keep all explanations for an observation.

Definition 16 *Let* $\mathcal{A}_{O,H}$ *be a diagnosis automaton, let* $F_{\max}$ *be the set of safe transitions of* $\mathcal{A}_{O,H}$, *and let* $F \subseteq F_{\max}$ *be a maximal connected subset of* $F_{\max}$. *A* module *for* F *is a triple* $Mod_F = (In_F, Out_F, T(Mod_F))$, *where* In_F *is the set of states which have predecessors out of* F *and* Out_F *is the set of states which have successors out of* F, *and* $T(Mod_F) \subseteq In_F \times Out_F$ *collects all pairs of states in* $In_F \times Out_F$ *for which there exists a path in* F

A module can be used to replace the safe transitions connecting unsafe parts of $\mathcal{A}_{O,H}$. One can easily prove that for any pair of paths ρ, ρ' and a module Mod_F such that ρ ends in a state s of In_F, ρ' starts in a state s' of Out_F, and $(s, s') \in T(Mod_F)$ then an observation $O' \sqsupseteq O$ has an embedding in $\rho.\rho_F.\rho'$ for every ρ_F path of F from s to s' if and only if $O' \setminus E_{\rho_F}$ and $\rho.\rho'$ satisfy the conditions of proposition 6.

Hence, a diagnosis automaton can be replaced by a set of unsafe transitions and a set of modules connecting them. Then the online diagnosis can be made as follows: Assume that O was observed and $\mathcal{A}_{O,H}$ was built (potentially with modules for $F_{\max}$).

First, we determine $F_{\max}$ as shown in section 5.1. The connected components that contain the initial state(s) can be forgotten or stored on disk, as they will not influence the construction of diagnosis. Then, for each remaining connected component F of $F_{\max}$, we create a new module Mod_F in $\mathcal{A}_{O,H}$, summarized as: for each state s of $\mathcal{A}_{O,H}$ such that there exists an outgoing transition in F and an ingoing transition not in F, s is an in-state of M_F. For each state s of $\mathcal{A}_{O,H}$ such that there exists an ingoing transition in F and an outgoing transition not in F, s is an out-state of M_F. We denote by $In(M_F)$ and $Out(M_F)$ its set of in-states and out-states respectively.

For each in-state s and out-state s' of Mod_F such that there exists a path with transitions of F from s to s', we create a module-transition from s to s'. We denote by $T(Mod_F)$ the set of module-transitions of M_F.

We denote by $S(F)$ the states to or from a transition of F. Now, we delete from the main memory (but keep on the hard disk) transitions of F, and states in $S(F) \setminus [In(M_F) \cup Out(M_F)]$.

Later, when we observe a new event e, we update the automaton (with modules) following propositions 7 and 9. The automaton with modules can be used exactly as $\mathcal{A}_{O,H}$. Here is the exact procedure.

Each transition $t = (n, E, g) \xrightarrow{M} (n', E', g')$, can be classified as follows:

a) – t has no predecessor containing unobserved events on process $\phi(e)$,
   – there exists an event f in E_M such that $\alpha(f) = \alpha(e)$ and $|h(E_{\phi(e)})| = |\{f' \in E_M \mid \alpha(f) \in \Sigma_O \wedge f' <_\phi (e) f\}|$ (f is the next event on process $\phi(e)$ in this path and it is compatible with e). Slightly abusing the notation, we write $h(E) = X$ instead of writing $h_{O,M_\rho}(E) = X$ for each path ρ that ends at transition t. Note that this has to be checked on one path ending at state (n, E, g) only, as once E is given, the embedding relation in E_M is unique. We will use this notation in the next items.
   – For each $e' <_{O'} e$, there exists $f' \in M$, such that $f' < f$, and either $e' \in g(\phi(f'))$ or $h(f) = e'$. (predecessors of f and predecessors of e are compatible)

b) – t has no predecessor containing unobserved events on process $\phi(e)$,
   – there exists an event f in E_M such that $|h(E_{\phi(e)})| = |\{f' \in E_M \mid \alpha(f) \in \Sigma_O \wedge f' <_\phi (e) f\}|$, but $\alpha(f) \neq \alpha(e)$ (The next observable event on process $\phi(e)$ is not compatible with e)

c) – t has no predecessor containing unobserved events on process $\phi(e)$,
   – there exists an event f in E_M such that $\alpha(f) = \alpha(e)$ and $|h(E_{\phi(e)})| = |\{f' \in E_M \mid \alpha(f) \in \Sigma_O \wedge f' <_\phi (e) f\}|$ (The next observable event on process $\phi(e)$ is compatible with e)
   – There exists $e' <_{O'} e$, such that for all $f' \in M$ with $f' < f$, we have $e' \notin g(\phi(f'))$. (predecessors of f and predecessors of e are not compatible)

d) – t has no predecessor containing unobserved events on process $\phi(e)$,
   – $|h(E_{\phi(e)})| = |\pi_{\Sigma_O}(M) \cap E_{\phi(e)}|$ (all observable events on $\phi(e)$ have already been observed). In particular, this means that there exists no event f in E_M such that $|h(E_{\phi(e)})| = |\{f' \in E_M \mid \alpha(f) \in \Sigma_O \wedge f' <_\phi (e) f\}|$

e) t is a maximal transition in $\mathcal{A}_{O,H}$ for process $\phi(e)$, i.e. it has no successor transition, and for every preceding transition $t_i = (n_i, E_i, g_i) \xrightarrow{M_i} (n'_i, E'_i, g'_i)$, all events on process $\phi(e)$ in M_i have been observed.

f) t is **not** the next transition containing unobserved events on process $\phi(e)$ along any path containing this transition,

g) t is a module transition (in particular, this means that all transitions appearing in the module are safe, so t can be ignored).

Case a) corresponds to paths which explains $O \cdot e$, without adding new transitions to $\mathcal{A}_{O,H}$. Cases b), c) correspond to paths which do not correspond to $O \cdot e$ (because the first observable on the process does not correspond to e (case b)), or because the causalities are not ensured (case c)). We can delete b and c type transitions from $\mathcal{A}_{O,H}$, as well as any path containing a transition of this kind, which is not an explanation for $O.e$

Case d) corresponds to transitions which observable events on process $\phi(e)$ have all been observed and mapped to an event of O. There is still one process containing events to be observed, as t is not safe.

Case e) corresponds to transitions which are at the end of a path ρ in which all events on $\phi(e)$ have already been observed. This path must be extended to find an explanation for e.

Case f) corresponds to transitions for which there exists a preceding transition still containing an unobserved event, hence no event of M should be mapped to e. However, these transitions can be used later to explain further events. Case f corresponds to the last transition of a path in which all observable events located on process $\phi(e)$ have been observed. This path must hence be extended to find new observable events on $\phi(e)$.

Case g) describes how to deal with module transitions. Theses transitions represent safe parts of paths and are just kept in memory to preserve connectivity in the diagnosis automaton. They can not be refuted nor provide an explanation for newly observed event e. They are simply ignored, and can even be erased if they are minimal in the diagnosis automaton.

Let T_a be the set of type a transitions. For every transition $t \in T_a$, we need to update t_a and all its successors. Let T_b and T_c be the set of type b and c transitions. We need to remove from $\mathcal{A}_{O,H}$ all paths containing a transition in $T_b \cup T_c$. Transitions in $T_d \cup T_f \cup T_g$ do not trigger any modification of $\mathcal{A}_{O,H}$. Finally, for every transition t_e of T_e, we need to extend the path ending in t_e to find an explanation for event e. If no extension exists, then all paths containing t_e must be removed from $\mathcal{A}_{O,H}$. This gives us the following algorithm:
For the sake of readability, we have split this algorithm into several phases that classify transitions, updates them, and extends e-type transitions. Most of these procedures (excepted the extension) can be achieved by a Depth First Search (DFS) on the existing summary. Furthermore, all these procedures can be grouped in a single exploration of $\mathcal{A}_{O,H}$.

Proposition 11 *Let $\mathcal{A}_{O,H}$ be a (summarized and abstracted) diagnosis automaton for an observation O, and let e be a new event. Then the size of the summarized and abstracted diagnosis $\mathcal{A}_{O.e,H}$ is in $O(|\mathcal{A}_{O,H}|+Ke.|\mathcal{P}|.|H|.2^{|\mathcal{P}|})$, where Ke is the number of type e transitions, and can be computed in $O(|\mathcal{A}_{O,H}|+Ke.|\mathcal{P}|.|H|.2^{|\mathcal{P}|})$.*

proof : Finding $F_{\max}$ is linear in the size of $\mathcal{A}_{O,H}$, as well as search for connected components. Removing transitions from $\mathcal{A}_{O,H}$ can be done with a complexity that depends on the set to be removed, which is necessarily of size lower than $|\mathcal{A}_{O,H}|$. Note that removing $F_{\max}$, T_b and T_c can be done at the same time. Finding accessible and coaccessible states can be done in at most $2.|\mathcal{A}_{O,H}|$. Dealing with transitions in T_a means a DFS in $\mathcal{A}_{O,H}$ (all updates due to transitions in T_a can be performed during the same exploration. The costly part can be to extend the maximal paths of $\mathcal{A}_{O,H}$, which can be done in

Algorithm 1 IncrementDiagnosis($\mathcal{A}_{O,H}$,O,e)

```
Compute $F_{\max}$
/* nb : $F_{\max}$ may contain module transitions */
Compute $C$, connected components of $F_{\max}$
for $c \in C$ do
    /* store $c$ on disk */
    $\mathcal{A}_{O,H} := \mathcal{A}_{O,H} \setminus c$
    if $c$ is not an initial component then
        $\mathcal{A}_{O,H} := \mathcal{A}_{O,H} \cup T(Mod_c)$
    end if
end for
$T_a = T_b = T_c = T_d = T_e = T_f = T_g = \emptyset$
for $t \in \mathcal{A}_{O,H}$ do
    class $t$ in $T_a, T_b, T_c, T_d, T_e, T_f$ or $T_g$
end for
$\mathcal{A}_{O,H} := \mathcal{A}_{O,H} \setminus (T_b \cup T_c)$
restrict $\mathcal{A}_{O,H}$ to transitions that are accessible from an initial state, and coaccessible from
sates of the form $(n, E_O, g)$ (DFS search).
for $t_a \in T_a$ do
    $t_a$ is of the form $t_a = (n, E, g) \xrightarrow{M} (n', E', g')$ and $f_a$ is the event mapped to $e$ in $M$
    Compute $t'_a = (n, E, g) \xrightarrow{M} (n', E \cup \{e\}, g'')$, where
    $g''(p) = g'(p) \cup \{e\}$ if there exists $f' > f_a$ such that $\phi(f') = p$
              and $g''(p) = g'(p)$ otherwise
    $\mathcal{A}_{O,H} := \mathcal{A}_{O,H} \setminus \{t_a\} \cup \{t'_a\}$
end for
Perform a DFS to update event set $E$ and function $g$ at each node reachable from a
transition in $T_a$.
for $t_e \in T_e$ do
    $t_e$ is of the form $t_e = s \xrightarrow{M} s'$
    /* Extend the path ending in $s'$ if possible */
    $X = Ext(s', e)$
    if $X \neq \emptyset$ then
        $\mathcal{A}_{O,H} := \mathcal{A}_{O,H} \cup Ext(s', e)$
    end if
end for
restrict $\mathcal{A}_{O,H}$ to transitions that are accessible from an initial state, and coaccessible
from states of the form $(n, E_O \cup \{e\}, g)$ (DFS search).
return $\mathcal{A}_{O,H}$
```

$Ke.|\mathcal{P}|.|H|.2^{|\mathcal{P}|}$. The size of the resulting summary increases the size of $\mathcal{A}_{O,H}$ by at most $Ke.|\mathcal{P}|.|H|.2^{|\mathcal{P}|}$ states. □

Note that for a diagnosis automaton of height h and and HMSC of degree d, we have that $K_e \leq d^h \leq |\mathcal{A}_{O,H}|$.

Corollary 2 *Computing online a summary for an observation O can be done in*

$$O\left(\sum_{i \in 1..|O|} |H|.(i-1)^{|\mathcal{P}|.|\mathcal{P}_{Obs}|} + d^{h(i-1)}.|H|.|\mathcal{P}|.2^{|\mathcal{P}|}\right)$$

where $h(i)$ is the maximal height of the diagnosis automaton built at step i.

Let us compare the respective complexities of the naive online diagnosis mentioned at the beginning of this section (that is $O(|H|.\sum_{i=1..|O|} i^{|\mathcal{P}|.|\mathcal{P}_{Obs}|})$) and this incremental diagnosis. At first sight, incremental construction of a diagnosis seems inefficient, as it has an overhead in $O(|H|.\sum_{i=1..|O|} d^{h(i-1)}.|H|.|\mathcal{P}|.2^{|\mathcal{P}|})$. This is due to the fact that, to compute a maximal safe part, one may have to extend all leaves of the previously built diagnosis, and that nothing guarantees that large subsets of transitions become safe at every construction step. Hence, the complexity above does not take into account the fact that at each step, a part of the diagnosis can be stored on disk and erased from memory. Note also that $d^{h(i)}$ is a rough upper bound on the number of paths to extend at step i, but that in practice, it is unlikely that all MSCs in $\mathcal{M}$ can be chosen from any state, or that all path need to be extended. Hence, the actual number of path in the summary at step i should remain lower than $d^{h(i)}$. Note also that $h(i)$ is necessarily lower than $|H|.i$. Last, we know that the incremental extension of the summary at step i can be done by studying only the unsafe part of the summary at step $i-1$. But there is no guarantee that this unsafe part remains bounded. However, if one can erase safe transitions at the same rate as new transitions are appended to extend the diagnosis, then there is a constant K such that throughout the computation, the unsafe part of the computed summary remains of height lower than K. Then the complexity of incremental diagnosis is in $O\left(|O|.(d^K + d^K.|H|.|\mathcal{P}|.2^{|\mathcal{P}|})\right)$.

Note that so far, even if $\mathcal{A}_{O,H}$ is an abstracted summary, there is no guarantee that the size of its unsafe part remains bounded. In the general case, the height of summaries remains bounded if all observed processes in the running application produce their observations at the same rate (i.e. there is no race between processes in the implementation), and if the arrival of observations to observers guarantees that each transition of the diagnosis automaton becomes safe after a finite number of observations. Last, notice that the diagnosis automaton obtained from a summary built online is exactly the automaton obtained in the offline setting, that consequently have the same number of states in $O\left(|H|.|O|^{|\mathcal{P}|.|\mathcal{P}_{Obs}|}\right)$.

Note however that in general nothing guarantees that an implementation is race free. Consider for example the HMSC of Figure 15. This model is composed of three MSCs, and the observation alphabet is $\Sigma_{obs} = \{a, a', b, c, d\}$. With this model, any path explaining an observation O has to remember $M_1.M_2^{|O|-1}$ as long as atomic action c has not been observed. If the sequence of observed events at step i is of the form form $a.(a'.b)^i$, then the unsafe part kept in memory by the algorithm is of the form $(n_0 \xrightarrow{M_1} n_1).(n_1 \xrightarrow{M_2} n_1)^i$, and the successive observations increase the size of the unsafe part of the computed summary if process P_3 is slower than all others. Hence, if $P1$ and $P2$ are faster than $P3$, or if $P3$ need more time than other processes to report occurrences of c, the unsafe part of the diagnosis automaton built from the observations may grow up to an arbitrary size.

In the sequel, we will show some syntactic conditions on H that ensure that for every sequence of observable events, incremental construction of diagnosis can be done with unsafe suffixes of bounded size in memory.

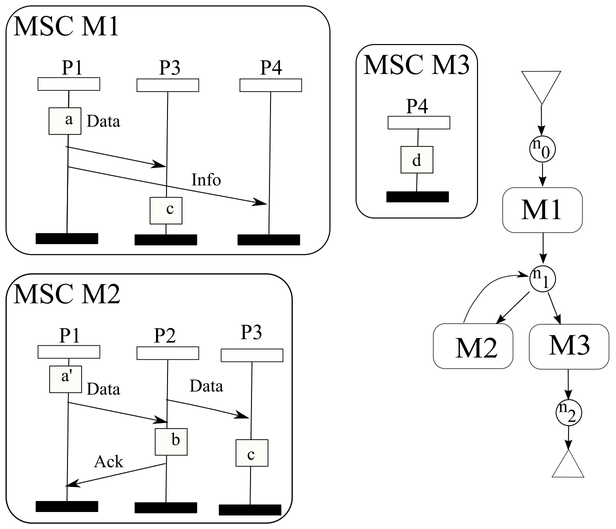


Fig. 15 An example HMSC in which online observation can grow unboundedly

5.4 Online diagnosis with finite memory

In this section, we address the problem of online diagnosis with finite memory. If Σ_{Obs} is located on a single process p, then the projection of an HMSC on a single p is a regular language, and we can perform diagnosis with a finite state automaton that monitors process p as in [24]. However, observing a single process is a very restrictive solution. A natural question is whether, for a given observation alphabet, some systems can be modeled by HMSCs for which diagnosis is always feasible with finite memory, independently from what is observed. The challenge is then to exhibit a subclass of HMSCs that ensures that the part of summaries that is kept in memory remains bounded. When a system can be modeled by a HMSC lying within this class of models, online diagnosis runs with finite memory. A first obvious candidate is the so-called syntactic class of *regular Message sequence charts* introduced by [1].

Definition 17 *Let M be a MSC. The* communication graph $CG(M) = (P, V)$ *is a directed graph such that :*

- $P = \{\phi(E_M)\}$ *is the set of processes that are active in* M,
- $(p, q) \in V$ *iff* M *contains a message exchange from* p *to* q, *i.e.* $V = \{(\phi(e), \phi(e')) \mid (e, e') \in \mu_M\}$

A HMSC H is called a regular HMSC *if and only if for every cycle ρ of H the communication graph $CG(M_\rho)$ is strongly connected.*

Alur et al. have shown [1] that for any regular HMSC $H = (N, \longrightarrow, n^i, \mathcal{M}, F)$, the set $Lin(H)$ forms a regular language. This language is recognized by an automaton $\mathcal{A}_{Lin(H)}$, of size at most in $O(2^{|\mathcal{P}|.|N|}.(b.|N|.|\mathcal{P}|)^{|\mathcal{P}|})$. The intuition behind regular HMSCs is that a process can not be too far ahead from other participants of a protocol, and has to wait for some "acknowledgement" all its actions in any iterative behavior. This way, a process p can not perform an arbitrary number of actions independently from the other processes, and can not either send and arbitrary number of messages without waiting for the reception of a message from the other processes. The whole linearization language of a regular HMSC H is recognized by a finite automaton $\mathcal{B}_H$, that memorizes the set of actions that have to be done by each process (we call the states of $\mathcal{B}_H$ *configurations* of H) to remain compatible with H. One can see configurations as a list of unexecuted parts of MSCs. The projection of linearizations of H on the observation alphabet Σ_{Obs} is also a regular language, that is recognized by an automaton $\mathcal{B}'_H$ which transitions are labeled by letters from Σ_{Obs}, and which states are a subset of configurations from $\mathcal{B}_H$.

However, observing a regular systems is not sufficient to ensure that diagnosis can be performed with finite memory, as the observation process itself can introduce some asynchronism between actual behavior and collected observations. Indeed all probes do not necessarily send their observations at the same speed. They can for instance be located at different places of a network, and very far from the diagnoser. Hence, even if all processes of the implementation are well synchronized, some processes might appear as ahead of the others in the observation.

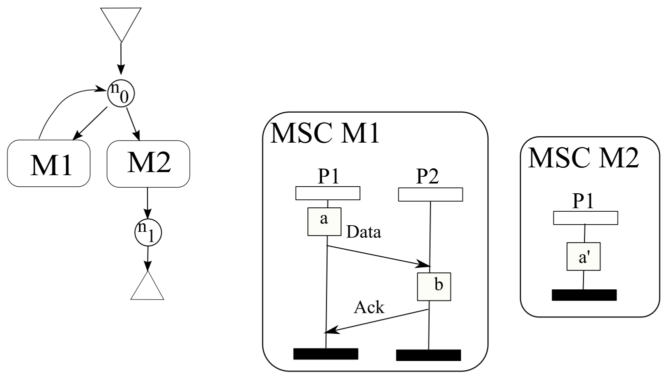
Consider for instance the example of Figure 16. The observable events are actions a and b, but the probe attached to process $P1$ needs at most 1 time unit to sent its observations, while the probe attached to process $P2$ needs at most 3 time units. At a given moment, the online observation of this regular system may exhibit much more a's than b's, as shown in the chronograms at the bottom of the figure. For instance, at time 6, the observation received at the diagnoser exhibits 3 occurrences of a and one b, even if in the corresponding execution, the difference between the number of a and b is lower or equal to 1. However, if the observation architecture guarantees that the delay on the observation is bounded w.r.t. the actual behavior, and if the processes of the system execute events with a known maximal rate, then online diagnosis with finite memory is achievable.

Theorem 4 *Let H be a regular HMSC, let Σ_{obs} be an observation alphabet. If every process in the observed system sends its observations with a bounded delay δ_{obs}, and an execution rate r, then online diagnosis for an observation O can be performed with memory at most in $O(NO.k.2^{|\mathcal{P}|.|N|}.(b.|N|.|\mathcal{P}|)^{|\mathcal{P}|})$, and in time lower than*

$$O(|O|.NO.k.2^{|\mathcal{P}|.|N|}.(b.|N|.|\mathcal{P}|)^{|\mathcal{P}|})$$

where $k = |\mathcal{P}|.\delta_{obs}.r$, and NO is in $O\left(\sum_{i\in 1..k} 2^{\frac{i^2}{4}.\frac{3i}{2}.\ln i}.|\Sigma_{obs}|^i\right)$

Proof: The main idea is to monitor the system with the finite automaton $\mathcal{B}'_H$ that describes the projection of all inearizations of H. However, the trick is to consume events of the observation O in sequence, that is choose online a linearization of O that is also a linearization of some behavior of H. States of $\mathcal{B}'_H$ describe configurations of H, that is the maximal node of H visited, and the part of each visited MSC that have not yet been executed. At each



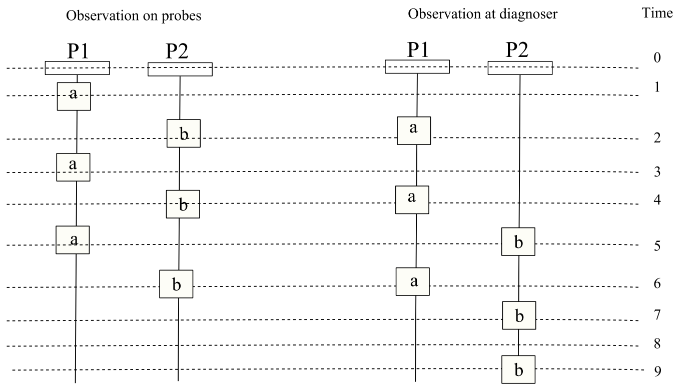


Fig. 16 Desynchronization of observation at probes and at diagnoser

step, the system can be in several of these configurations. When a new event is observed, for a given state s, one must decide whether this event is compatible with the configuration, and move to a new set of target configurations. If the observed event e if not fireable from state s, then two solutions are possible: either e can not be the next event observed on process $\phi(e)$, and the assumption that the system was in state s was wrong. In such case, all sequences of transitions ending at s at former step must be discarded. The other possibility is that according to the configuration in s, e has a predecessor f, that must be observed. However, observation of f may arrive later than that of e. Note that according to our observation semantics, f and e can be observed as concurrent events, or we can have $e \leq f$, but not the converse. Hence, one may have to wait up to δ_{Obs} time units to consume f, then e. In the meantime, the observer may observe the arrival of $\delta_{Obs}.r$ new events. If f is not observed after this delay, then the linearization ending in state s was not an explanation for the observation, and this state can be forgotten, or marked as wrong. Note however that as H is regular, processes have to wait for one another if they are active within the same loop. Hence, inside a single loop β, a group of processes may force the system to recall at most $\delta_{Obs}.r.|\mathcal{P}|$ events and can then consume one minimal event every $1/r$ time unit. Similarly, if the system has entered a behavior described by another loop β' of H, that does not involve processes participating in β. However, events from this loop must be fireable from s, as both loops are concurrent, or again, s must wait to consume events, but need only to recall a bounded number of events. Note that there can be at most $|\mathcal{P}|$ such concurrent loops. So, when n events have arrived at the observer, the explanations (linearizations) computed so far may explain the whole observation or only up to some $k \leq n$, with $k > n - |\mathcal{P}|.\delta_{obs}.r$ and store events. Hence, at a given moment, one may have to remember up to $NO.2^{|\mathcal{P}|.|N|}.(b.|N|.|\mathcal{P}|)^{|\mathcal{P}|}$ states, where NO is the number of observations of size at most $|\mathcal{P}|.\delta_{obs}.r$. Kleitman and Rotschild [14] gave a bound of $2^{\frac{n^2}{4}.\frac{3n}{2}.\ln n}$ for the number of partial orders of size n. We must in addition label these orders, i.e. NO is in $O\left(\underset{i\in 1..k}{\Sigma} 2^{\frac{i^2}{4}.\frac{3i}{2}.\ln i}.|\Sigma_{obs}|^i\right)$, for $k = |\mathcal{P}|.\delta_{obs}.r$ □

From Theorem 4, it is straightforward to design a diagnosis algorithm. We start the diagnosis from the initial state of $\mathcal{B}'_H$. we keep sates together with the step at which they were produced (i.e. pairs of the form (s,i), where i means that s is remembered since i^{th} observed event. When a new event e arrives at a date d(e), for each state kept in memory:

– a transition by $\alpha(e)$ is fireable from state s to state s in $\mathcal{B}_H$. Then store on disk transition $(s, i) \xrightarrow{\alpha(e)} (s', i + 1)$. If (s, i) was associated with a partially ordered set M of memorized events, then repeat the transition operation following linearizations of M.
– no transition by $\alpha(e)$ is fireable from state s. If there is no reachable transition from s by $\alpha(e)$, then s is marked as wrong state. If e needs to wait for a predecessor event to be fireable, then e is kept in memory with its date of observation $d(e)$.
– every state with memorized event with date lower than $d(e) - \delta_{obs}$ is declared wrong.

At the end of the observation, one can rebuild all transitions, and discard the path that lead to a wrong state. Note that unlike the offline or incremental diagnosis proposed before, we explore linearizations of H, which can be costly. A second remark is that if we run the incremental diagnosis of section 5.3 with a regular HMSC, we might be unable to forget safe parts.

Consider for instance the regular HMSC of Figure 17, and let us set as observation alphabet $\Sigma_{Obs} = \{a, b, c\}$. When an observation that contains as many occurrences of a and b, but no occurrence of c, one can not decide whether MSC M_2 occurred or not. Hence, the summary built from such observation has the shape of the automaton of Figure 18 (for the sake of clarity, we omit E and g in states). Transitions labeled by M_1 can be replaced by an equivalent module, as all events in M_1 are safe. However, the connectivity provided by these modules must be preserved, and none of them can be removed from the summary. Hence, as each M_1 transition is equivalently replaced by a module, the size of the summary is not decreased. This situation holds for observations of any size that do not contain an occurrence of event c. Hence, even if we

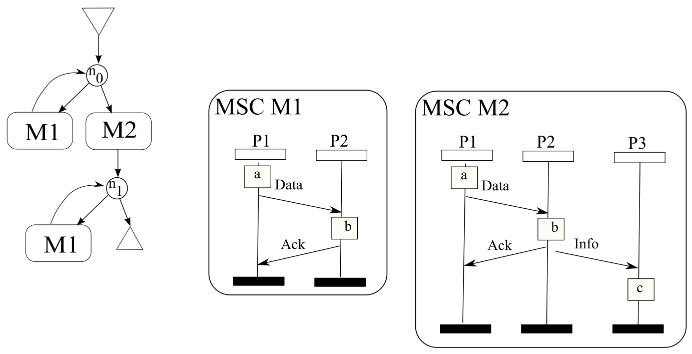


Fig. 17 A regular HMSCs that may generate small modules

have guarantees on delays and rates of the observed system and H is regular, the incremental diagnosis may have to keep an infinite summary in memory.

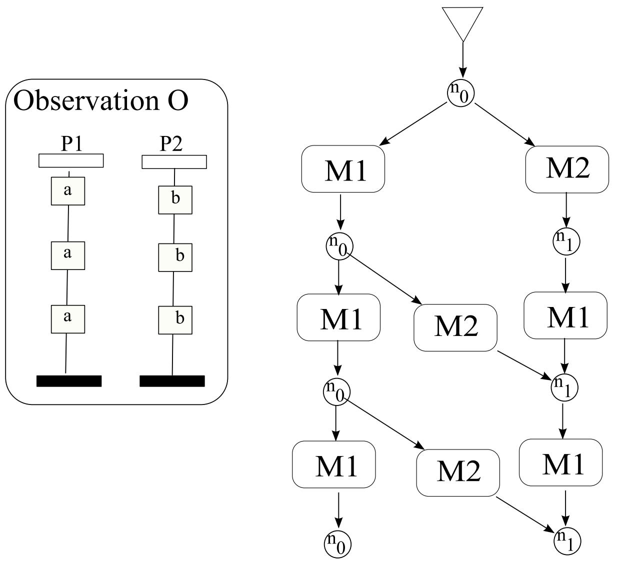


Fig. 18 The diagnosis for an observation O.

5.5 Online detection of existence

When the existence problem is solved offline, it simply consists in building the diagnosis automaton as described in section 4 , and verify that the language of this automaton is not empty. For the online detection, the fact that we do not have to output a diagnosis simplifies the process, as the safe parts of the diagnosis that are built online need not be stored, and can just be forgotten. This leads to the algorithm 2.

Algorithm 2 OnlineExistence()

```
Continue := true ; $O = \emptyset$
while Continue = true do
    receive observation $\{e\}$
    continue := false
    /* progress diagnosis */
    $\mathcal{A}_{OH}$ := IncrementDiagnosis($\mathcal{A}_{O,H}, O, E$)
    if $\mathcal{A}_{OH} = \emptyset$ then
        Continue := false
    else
        $O = O \cup (e, \{e\} \times Pred(e))$
        /* add $e$ to the observation */
    end if
end while
Output “No explanation in $H$ for current observation.
```

6 An application of diagnosis for Anomaly Detection

The diagnosis framework shown in previous sections is originally designed to help debugging a distributed application when a fault has occurred. In this section, we address another possible application of diagnosis for security. In a diagnosis framework, the model represents the expected behavior of the system. We have already mentioned that the diagnosis obtained from an observation may produce an empty set of explanations. In a debugging context, this can be bothering, and means that our model is not complete enough to provide explanations for a given observation. If on the contrary, we consider that our model is a complete representation of the normal use of a system, then, finding no explanation for a given observation means that the currently observed execution **is not** a normal behavior, and that our system may have security problems: it is attacked by an intruder, some process has been corrupted, ... We will show in this section how diagnosis can be used in the context of *anomaly detection*.

Since the 80's several intrusion detection systems (IDS) have been proposed. An important class of these systems is “signature-based”, that is recognizes patterns from a collection of known attacks. These IDS can recognize attacks in a single frame from packet headers, or in a session that is a sequence of packet exchanges between two given IP addresses. The techniques used to recognize an attack range from Bayesian networks, statistical methods, fuzzy inference systems, etc. to data mining techniques. Usually, IDS try to classify a situation according to an attack nomenclature, or as normal if the observed behavior is too far from the criteria that characterize an attack. To train IDS, huge databases of attacks have been collected [20,6].

However, these IDS mechanisms suffer several weaknesses. First of all, as IDS are trained from datasets, they can not discover novel attacks. For this reason, a new complementary solution called *anomaly detection* has been proposed. Anomaly detection again rely on comparison of observations, but this time the monitoring systems that are used compare the observation with a description of a normal situation. The assumption is that when an attack occurs, it usually exploits weaknesses of a system, that are rarely used in normal conditions. This is peculiarly true for denial of service attacks, where the rate of some requests suddenly becomes unusual. Hence, a detection of some unusual behavior in a system is assumed to be a potential attack, and raises an alarm. The main challenge here is to establish a profile of normal behaviors. This can be done in the same ways as for IDS, using statistical techniques [22], or with a specification-based approach. [15] proposes a logical framework to define normal behaviors, and [25] proposes a definition of normal behaviors using extended finite state machines. Detection of anomalies is then performed by comparing a linearization of an execution with the descriptions of correct behaviors.

Another weakness of session or packet based IDS is that they can not deal with attacks that involve several users or processes in a system. This is the case for example of covert channels, that use legal access to a system to create illegal flows of information between unauthorized peers. These covert channels, however, often use resources repeatedly and in a non-standard way. We then need techniques that detect attacks involving an arbitrary number of processes. Existing specification based anomaly detection frameworks such as [25] can take this into account, but rely on interleaved representations of behaviors, which increases the size of models. The executions of distributed systems can be represented as partial orders, hence computing or representing interleaved models is clearly not needed, and slows down the detection of anomalies. Several surveys on IDS and anomaly detection have already been published. We refer interested readers to [2,12,13] for more information.

We propose a scenario-based anomaly detection framework that uses partial order diagnosis techniques to detect abnormal behaviors. The advantages when using scenarios are manifold. First, scenarios are partial order models, and can avoid costly interleaved representations. Second, scenario are widely used to define systems requirements. These requirements can serve as a starting basis to create larger models of legal behaviors. The main idea behind the solution proposed is to observe the running system, and to compare the observation with a set of predetermined "standard" behaviors, defined as a (potentially infinite) collection of partial orders. When an observed run can not be described as a superposition of standard executions, then it is considered as suspect. This can be compared with diagnosis techniques, and we will show in the sequel that scenario-based anomaly detection can be brought back to the existence problem. Our detection framework analyzes behaviors involving multiple exchanges among several processes, without computing an interleaved representation of the legal behaviors nor of the observation.

6.1 Monitoring architecture

The anomaly detection framework proposed hereafter relies on the diagnosis techniques proposed in section 4 and 5, and on an architecture that collects a subset of what occurs in the system, and compares the observation to the HMSCs defining normal behaviors.

The framework we consider is a distributed system, composed of several sites (or processes) $\mathcal{P} = P_1, \dots, P_n$, providing distributed applications $\mathcal{D} = A_1, \dots, A_k$ to a set of users $\mathcal{U} = U_1, \dots, U_q$. This system is monitored by inserting probes on each site as described in section 1, with the only difference that the diagnoser will compare observed executions with *several models*.

Each user can run several applications of the system, according to a predefined policy. Whenever a user $U_i, i \in 1..q$ uses an application $A_j, j \in 1..k$, we depict the normal use of application j by user i as a HMSC H_{ij}, and define an observation alphabet Σ_{ij}. We set $H_{ij} = (N_{ij}, \longrightarrow_{ij}, n^i_{ij}, F_{ij}, \mathcal{M}_{ij})$, where all MSCs in $\mathcal{M}_{ij}$, are defined over a subset of $\mathcal{P}_{ij} = U_i \cup \mathcal{P}$. More intuitively, the interactions depicted do not involve other users of the system. We also require that $\Sigma_{i,j} \cap \Sigma_{k,l} = \emptyset$ for every $(i, j) \neq (k, l)$. This may seem restrictive, but if we consider that all communications and events during a sessions are tagged with an unique session number, this property is immediately met. Note also that we could define more general HMSCs involving several users and several applications without changing the formal techniques described hereafter. However, we feel that such models would be much more difficult to design.

The system is instrumented to detect the occurrence of some events with signature in $\bigcup_{i \in 1..q, j \in 1..k} \Sigma_{ij}$ and to send them to a centralized diagnoser that logs all events. This is the usual diagnosis architecture defined in previous sections. Here again, the observation mechanisms can provide some causal ordering among events, and events located on a given process are totally ordered. The observation collected is compared with the models of legal use of all applications by the diagnoser. This monitoring architecture is depicted in Figure 19. The logged file can be seen as an observation.

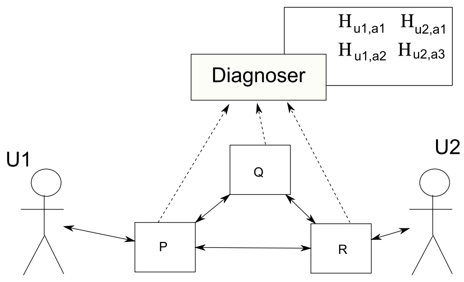


Fig. 19 Architecture of the anomaly detection framework

6.2 Anomaly detection with Diagnosis techniques

The anomaly detection framework proposed in this paper relies on the diagnosis techniques presented in the previous sections. Note however that observations of distributed systems will mix uses of several applications by several users, and we need to recover the order associated to each pair $(user, application)$

that is contained in O, but forget the causal ordering that is due to messages exchanges from other users and applications. This is not captured by the definition of projection, and we need to define a new restriction operation to separate the observation of different sessions:

Definition 18 *Let* $O = (E_O, \leq_O, \alpha_O, \mu_O)$ *be an observation defined over a set of processes* $\mathcal{P}$ *and over an alphabet* $\Sigma = \Sigma_{ij} \cup \Sigma'$*. The* restriction *of* O *to* Σ_{ij} *is an observation* $R_{\Sigma_{ij}}(O) = (E_{O'}, \leq_{O'}, \alpha_{O'}, \mu_{O'})$ *such that* $E_{O'} = E_O \cap \alpha^{-1}(\Sigma_{ij})$*,* $\mu_{O'} = \mu_O \cap E^2_{O'}$ *and* $\leq_{O'} = \big(\{(e,e') \in E^2_{O'} \cap \leq_O \mid \phi(e) = \phi(e')\} \cup (E^2_{O'} \cap <_O)\big)^*$

Let us now show how the existence problem described in section 4.1 can be used for anomaly detection. The main idea in this approach comes from the fact that diagnosis may fail when the model of our system does not provide executions that are compatible with an observation. In a distributed diagnosis, failure of diagnosis can be much faster than correct diagnosis, as no explanation exists for the considered observation as soon as a sub-problem can not be solved. Now, if we consider that a HMSC model provided to the diagnoser represents *all* legal behaviors, diagnosis failure can be interpreted as the occurrence of an illegal behavior.

We propose a partial order anomaly detection framework based on the architecture and on the diagnosis techniques of sections 4 and 5. This detection can be performed either:

- offline, that is after recording an execution, the anomaly detection algorithm is run to discover whether this execution contains an attack
- online, that is a monitoring systems analyzes current execution and raises a warning as soon as an anomaly is detected.

An advantage of the proposed techniques is that there are no wrong positives. Another advantage is conciseness of the models and of anomaly diagnosis: legal behaviors are given in terms of HMSCs, i.e. the interleaved representation of legal behaviors is never computed. We now describe these two techniques, and compare their advantages and drawbacks.

Based on the observation architecture described in section 4, we can define an offline detection framework for unusual behaviors. First of all, we can notice that if we use a HMSC H_{ij} to describe the behaviors attached to user u_i and to the system when running application j, we need to allow this user to run this application several times. Furthermore, an attack is not necessarily contained in a single use of an application. We then have to compare several successive (mis-)use of an application with normal use. This can be defined by computing a cyclic version of $H_{ij} = (N_{ij}, \longrightarrow_{ij}, n^i_{ij}, F_{ij}, \mathcal{M}_{ij})$, denoted by H^*_{ij}, and defined as $H^*_{ij} = (N_{ij}, \longrightarrow'_{ij}, n^i_{ij}, F_{ij}, \mathcal{M}_{ij})$, where

$$\longrightarrow'_{ij} = \begin{array}{l} \longrightarrow_{ij} \setminus \{(n, M, n') \mid n' \in F_{ij}\} \\ \cup \{(n, M, n^i_{ij}) \mid \exists (n, M, n'') \in \longrightarrow_{ij} \wedge n'' \in F_{ij}\} \end{array}$$

In a more intuitive way, in H^*_{ij}, transitions to final states are redirected to the initial state. We will consider that there is an anomaly when an observed behavior can not be explained as a mix of all legal behaviors defined by HMSCs in $\{H^*_{ij}\}_{i\in 1..q, j\in 1..k}$. This is usual for anomaly detection, as we can consider that an attack exploits unknown (and then unused) weaknesses of a system, and furthermore do not necessarily have enough knowledge of the system to generate an attack while behaving as in a normal session.

Definition 19 *Let $U_1, \dots U_q$ be a set of users of a system composed of processes $P_1, \dots, P_n$ and applications $A_1, \dots A_k$. Let O be the observed behavior of the system. Then U_i has an* unusual behavior *in O when using application A_j if $R_{\Sigma_{ij}}(O)$ has no explanation in H^*_{ij}. An observed behavior contains an* anomaly *if and only if at least one user $U_i, i \in 1..q$ has an unusual behavior when using an application $A_j, j \in 1..k$.*

Theorem 5 *Let $\{H_{ij}\}_{i\in 1..k, j\in 1..q}$ be a set of normal behaviors of a system composed of q users, k applications, and n processes. Then, anomaly detection in an observation O can be performed in $O(k.q.h.|O|^{(n \cdot p_{obs})})$, where h is the size of the largest HMSC in all H_{ij}'s and p_{obs} is the maximal number of process observed in all H_{ij}'s.*

Proof: Simple corollary of theorem 1. Anomaly detection is simply the iteration of the existence problem for all models H^*_{ij}, and observations that are projections on the alphabet Σ_{ij}. Every H_ij^* is defined aver a set of processes of size lower or equal to n, plus an user. Then, a diagnosis has to be performed for every H^*_{ij}, that is at most $q.k$ times, and for a restriction of the observation, that is an observation of size lower than $|O|$. □

We have assumed for convenience and efficiency that all observation alphabets were disjoint. Within this setting, the conclusion of the analysis is obvious: when all restrictions $R_{\Sigma_{ij}}(O)$ have their explanation in H^*_{ij}, no alarm is raised, and when a single projection have no explanation, an alarm must be raised. Hence, if we consider that observations are faithful, that is all observed events really occurred, no observation is lost by the supervision architecture, and the order among them is contained in the order of the execution, then we can not have wrong positives: when an alarm is raised, the actual execution that has produced the observation is not a mix of MSCs provided by the models. Note however that wrong negatives can still occur. This is not surprising, as observations only record a subset of all events that have occurred during an execution, and similarly only a subset of causal ordering among the observed events. Hence, an observation might be compatible with some MSC generated by each model, but not the execution that lead to this observation. Hence, the causal ordering among events in an execution might contradict any explanation provided by the models (it may contain different events, and different causal ordering among observed actions), but the recorded information might still allow for some explanations.

Remark 3 The disjoint alphabets assumption can be relaxed, but forces considering all possible partitions of O, that is assign to each event of O a pair i, j of user and application. This means that in the worst case (when all Σ_{ij}

are equal), we have to apply diagnosis techniques to up to $(q.k)^{|O|}$ *different partitions of the observation. The exponential blowup is not the only problem with overlapping observation alphabets. If none of the possible partitions shows unusual behaviors, then the observation corresponds to an interleaving of projections of normal executions. If some (but not all) partitions exhibit an anomaly, then two possible interpretations can be considered: an alarm must be raised, as there is a possibility of abnormal behavior, or conversely, as at least one partition of events allows for an anomaly free interpretation of the observation, and the behavior observed should be considered as normal.*

Offline detection can be used when something went wrong in a system, to make sure that the reason for a failure, for data corruption, or something bad that occurred is not due to an attack. However, detection mechanisms find their full interest when they can be used online to monitor ongoing executions. Online detection mechanisms must raise alarms when an anomaly is detected. Then a supervisor, that might be an automatic process or a human operator has to analyze the threat and react accordingly. The decision that follows an alarm depends on the analysis performed by other detection mechanisms, on the severity of the supposed attack, but also on the security level that one wants to provide for a system, and may range from closing a session, banishing an user or an IP address from the system, to switching off the whole system.

Similarly to offline anomaly detection, online anomaly detection is an adaptation of the online existence problem. The main difficulty here is to maintain several copies of the online existence algorithm (one per H^*_{ij}, and to feed these monitors with the correct observed events. In a setting where all observation alphabets are disjoint, this is not a problem. In case these alphabets are not disjoint, the solution consists in assigning an observed event to a pair (user,application) and create a copy of all diagnosers for every possible assignment. However, the cost of this solution is rapidly prohibitive.

7 Conclusion

We have proposed a centralized offline and online diagnosis framework based on scenarios. The diagnosis problem can be easily split into sub-problems to speed-up the algorithms. An easier diagnosis related problem called the existence in NP-complete. We have shown that diagnosis can be computed offline from a definitive observation, or on the fly. An efficient online solution is to compute a summary of the diagnosis, and work only on the (unsafe) part of the summary that is needed to continue explanations. However, in the general case, this unsafe part is not bounded. We have shown some syntactic restrictions on HMSCs and on the observed system that allow for incremental diagnosis with finite memory. However, this result applies only if the communication of observations from probes to diagnoser does not let a process produce observable events much faster than the others (this phenomenon is mainly due to communication delays and execution rates). As the considered models for finite memory are regular HMSCs, processes must wait for each other in loops, and the corresponding consumption of observed events by the diagnoser almost follows the order of production. We also need to ensure that only a bounded number of new observations can occur between the moment a probe observes an event, and the moment this observation is used on the diagnoser. This is guaranteed by the bound on communication delays from the system to the diagnose, and by imposing a maximal rate to each process. This could also be ensured by adding some time information (for instance duration of events and communications) to HMSCs. In a timed context, we think that a bound on the size of the unsafe part might be easy to obtain for a larger class of HMSCs if we can ensure that a bounded number of observable events can be produced by each process during a time unit - this hypothesis is close to the non-zeno property that is frequently used in timed automata, see [26] for instance.

Another interesting extension of this work would be to consider diagnosis from more powerful scenario models such as Causal MSCs [8]. Indeed, MSCs do not allow for the design of behaviors such as sliding windows. This can be considered as a limitation, as these behaviors are quite frequent in actual protocols. However, extending the scenario model inconsiderately could rapidly make diagnosis an undecidable problem (diagnosis is not decidable for communicating automata for example).

The diagnosis techniques proposed in this paper rely on behavioral models that are supposed close enough to the behaviors of an implementation. However, some discrepancies may still exist between the model and the set of behaviors exhibited by a running system, and our diagnosis algorithm may return an empty set of explanations. An interesting idea is to propose a diagnosis that returns the closest explanations found instead of an exact one, or the set of explanations of the largest prefixes of the observation for which an explanation exists. This would help finding the discrepancies between the model and the running system. Nevertheless, the fact that diagnosis may fail can be used for security applications. We have depicted an anomaly detection framework that detects an anomaly when the existence problem has a negative answer.

This framework differs from usual techniques relying on Bayesian networks, and from specification based techniques relying on interleaved models. An alarm is raised as soon as an observed behavior finds no explanation in the partial order models of usual utilization. This framework could be easily extended to take into account combinations of verdicts, integrate undesired behavior (known attacks), or allow some differences between the observation and the explanations provided by models, and raise alarms only when there is a certain distance with expected behaviors is exceeded. So far, we have only considered anomaly detection, but the proposed framework clearly applies to attack detection. It is well known that a large percentage of security attacks use security flaws that were made public, and for which solutions were also published (see for instance [3] for a list of vulnerabilities in TCP/IP). Such attacks can be described as finite scenarios. The challenge is then to decide whether an observation coincides in some way with an attack scenario.

At last, one noticeable fact is that in all the proposed solutions of the paper, the costly and sometimes infinite interleaved representation of HMSCs is never computed. This shows that it is feasible for some applications such as diagnosis to work almost entirely with partial order models.

References

1. R. Alur and M. Yannakakis. Model Checking of Message Sequence Charts. In *Proceedings of CONCUR'99*, LNCS 1664, pages 114–129, 1999.
2. S. Axelson. Intrusion detection systems: A taxomomy and surveytechnical report. Technical Report 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, 2000.
3. S. M. Bellovin. Security problems in the tcp/ip protocol suite. *SIGCOMM Comput. Commun. Rev.*, 19(2):32–48, 1989.
4. A. Benveniste, E. Fabre, C. Jard, and S. Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727, May 2003.
5. Jan A. Bergstra and Jan Willem Klop. Algebra of communicating processes with abstraction. *Theor. Comput. Sci.*, 37:77–121, 1985.
6. DARPA. Intrusion detection dataset. http://www.ll.mit.edu/IST/ideval/data/data_index.html, 2000.
7. C. Fidge. Logical time in distributed computing systems. *Computer*, 24(8):28–33, 1991.
8. Thomas Gazagnaire, Blaise Genest, Loïc Hélouët, P. S. Thiagarajan, and Shaofa Yang. Causal message sequence charts. In *CONCUR*, pages 166–180, 2007.
9. B. Genest, D. Kuske, and A. Muscholl. A kleene theorem and model checking for a class of communicating automata. *Information and Computation*, 6(204):920–956, 2006.
10. L. Hélouët, T. Gazagnaire, and B. Genest. Diagnosis from scenarios. In *Workshop on Discrete Event Systems, WODES'06*, 2006.
11. ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)*. ITU-TS, September 1999.
12. A.K. Jones and R.S. Sielken. Computer system intrusion detection: A survey. Technical report, Dept. of Computer Science, University of Virginia, 1999.
13. P. Kabiri and A.A. Ghorbani. Research on intrusion detection and response: A survey. *International Journal of Network Security*, 1(2):84–102, 2005.
14. D.J Kleitman and B.L Rotschild. Asymptotic enumeration of partial orders. *Transactions of the American Mathematical Society*, (205):205–220, 1975.
15. C. Ko, M. Ruschitzka, and K.N. Levitt. Execution monitoring of security-critical programs in distributed systems: A specification-based approach. In *IEEE Symposium on Security and Privacy*, pages 175–187, 1997.
16. F. Mattern. On the relativistic structure of logical time in distributed systems. *Parallel and Distributed Algorithms*, pages 215–226, 1989.
17. A. Muscholl. Matching specifications for Message Sequence Charts. In *FoSSaCS'99*, LNCS 1578, pages 273–287, 1999.
18. A. Muscholl and D. Peled. Message sequence graphs and decision problems on mazurkiewicz traces. In *MFCS*, pages 81–91, 1999.
19. A. Muscholl, D. Peled, and Z. Su. Deciding properties for message sequence charts. In *FOSSACS'98*, pages 226–242. Springer-Verlag, 1998.
20. University of California. Kdd cup 1999 data, 1999.
21. OMG. Uml superstructure specification, v2.0. OMG Document number formal/05-07-04, 2005.
22. O. Salem, S. Vaton, and A. Gravey. An efficient online anomalies detection mechanism for high-speed networks. In *MonAM'07*, 2007.
23. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, 1996.
24. F.B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.

25. R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: A new approach for detecting network intrusions. In *9th ACM conference on Computer and communications security*, 2002.
26. S. Yovine. Model checking timed automata. In *European Educational Forum: School on Embedded Systems*, pages 114–152, 1996.