Diagnosis from Scenarios

Loïc Hélouët, Hervé Marchand, Blaise Genest, Thomas Gazagnaire

▶ **To cite this version:**

Loïc Hélouët, Hervé Marchand, Blaise Genest, Thomas Gazagnaire. Diagnosis from Scenarios. Discrete Event Dynamic Systems, Springer Verlag, 2014, 24 (4), pp.353-415. 10.1007/s10626-013-0158-2 . hal-00879441

HAL Id: hal-00879441

https://hal.inria.fr/hal-00879441

Submitted on 3 Nov 2013

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

Diagnosis from Scenarios

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. Systems are modeled using High-level Message Sequence Charts (HMSCs), and the diagnosis is given as a new HMSC, which behaviors are all explanations of the partial 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 the memory used can be bounded. This allows us to give a complete diagnosis framework for infinite state systems, with a strong emphasis on concurrency and causal ordering in behaviors. 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

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
OCamlPro SAS, Gif-sur-Yvette, FRANCE

need to obtain information on what led to 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, 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. The choice of an appropriate subset of observable events influences accuracy and efficiency of diagnosis, and should be seen a part of the design of a complex system.

Several techniques are frequently called "diagnosis" while addressing different goals. Any technique that provides online or offline information on a system to a supervisor can be called diagnosis, but we will focus more precisely on one of them, namely history diagnosis. *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 solve different problems: fault diagnosis detects if a fault has occurred (and very often *which* fault occurred), while history diagnosis provides a set of 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 [7]. 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 allowing to model distributed systems working on different processes [18]. 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.

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 HMSCs 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 new model based on O and H that contains all possible explanations for observation O provided by H.

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 [24,12]. 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, we can reliably use the fact that an event did not appear in an observation to find or rule out explanations.

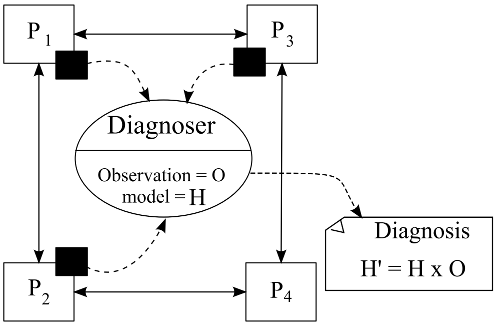

Fig. 1 Scenario-based diagnosis framework

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. Communication links 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 occurrences 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 what occurs on a process is monitored. The diagnoser receives individual event reportings from probes, and builds an observation O from the received information. It then uses a model (in our case an HMSC H) that describes all possible behaviors of the system (or at least a reasonably large subset of them) to output a new model (the *diagnosis*) that defines the set of all executions of H that are consistent with the observation O (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 modeled as an HMSC to explain a particular observation O. The explanation produced is a generator of all executions of the 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 diagnosis algorithm.
- The second result of this paper is that a global explanation can be reconstructed from local diagnosis performed for each pair of processes. 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. A last step combines the local explanations to produce a global explanation. This also opens the way for distributed diagnosis algorithms.
- 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 some sufficient syntactic restrictions to perform online diagnosis with finite memory.
- The fourth part of this paper shows an 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 existence problem. When no explanation of an observation can be found in a model, then the observed behavior is considered as illegal. witnesses an anomaly. This anomaly detection framework uses our diagnosis algorithms, and can then be used in an offline or online setting.

The major contribution of this paper is to propose a history diagnosis framework using infinite state models. This paper is an extended version of a preliminary work published in [17]. It is organized as follows: section 2 gives a taxonomy of diagnosis-related problems (namely fault diagnosis, history diagnosis and existence problems) and compares the approach of this paper with former works. Section 3 introduces the scenario language used, and section 4 introduces the formal definition of observations and explanations. Section 5 defines the main algorithms for diagnosis, and shows how to retrieve explanations in a decentralized framework. Section 6 shows how to adapt offline algorithms to perform online detection. Section 7 shows the application of diagnosis to anomaly detection. Section 8 concludes this work.

2 A taxonomy and state of the art of diagnosis related problems

In this section we formalize three problems related to observation and monitoring of systems, namely diagnosis, history diagnosis, and the existence problem. We also list some existing approaches for diagnosis and history diagnosis. The initial common assumptions for these three problems are that:

- We monitor a system composed of a set of independent processes $\mathcal{P} = \{P_1, \dots P_k\}$ that communicate asynchronously. This system does not necessarily have a finite number of states. Some processes in the system are instrumented (this can be done by inserting code in the process) to report some interesting events: message arrivals, sendings, exceptions,... Hence, we observe a modified version of the original system $\mathcal{P} = \{P'_1, \dots P'_k\}$, in which each P'_i is either the original process P_i or an instrumented version. In addition to the instrumented processes, the system is equipped with a particular process D called a *diagnoser*. Each P'_i produces observations and report them to the diagnoser. Observed events are occurrences of actions belonging to an observation alphabet Σ_{obs}. The more general setting for event reporting is that an instrumented processes P'_i send asynchronously to the diagnoser D an indication that some action in Σ_{obs} has occurred, along with a sequence number that is maintained on P'_i. This message sending to D as well as the sequence number management is part of the code inserted in P_i. The overall observation collected at diagnoser D from instrumented processes can be remembered as: a sequence of actions (a word from Σ^*_{obs}) ordered according to the notifications received by the diagnoser, a Parikh vector (a function $\Psi : \Sigma_{obs} \to \mathbb{N}$) counting the number of occurrences of each observable action, or as a partially ordered set of observable events. This latter is the setting that will be used in this paper, and we describe it more precisely in section 4. In the sequel, we will denote by O such an observation. As an observation records information on observable events, it is almost (up to some missing causal relations) the projection $\Pi_{obs}(R)$ of a run R of the system on its observable events.
- the known executions of the running system are represented by a model M. This model can be a set of logical formulae, a finite state machine, a Petri net,... The model M can describe behaviors as sequences of actions from a fixed alphabet $\Sigma \supseteq \Sigma_{obs}$, or in the case of HMSCs as partial orders. The set of all behaviors of M is called the *language* of M, and denoted by $\mathcal{L}(M)$. Some non-observable events called *faults* represent failures of the system. The model M is not always an exact representation of the running system. One reason is that full knowledge of the system's behaviors may not be available. Another reason is that the considered system *can not* be represented using the chosen formalism. Note also that all diagnosis problem may become undecidable if the chosen model is too powerful. For instance, one can show that diagnosis is undecidable for communicating Finite State Machines as a consequence of undecidability of reachability for this model [6]. Hence, the model M may only give an approximation of the actual behaviors of the running system. As a consequence, one may

obtain an observation O of the running system such that no execution of $\mathcal{L}(M)$ produces O when executed. We however suppose that the behaviors of the model and those of the running system are sufficiently close to ensure that diagnosis is significant, that is if we call $\mathcal{L}(\mathcal{P})$ the set of executions of the running system, we have $\frac{|\mathcal{L}(\mathcal{P})\setminus\mathcal{L}(M)|}{\mathcal{L}(M)} \sim 0$ and $\frac{|\mathcal{L}(M)\setminus\mathcal{L}(\mathcal{P})|}{\mathcal{L}(\mathcal{P})|} \sim 0$.

The objective of *fault diagnosis* is to detect, from an observation O and a model M whether a fault has occurred. We can formalize the question as " given an observation O and a model M, is there a run containing a fault f in $\Pi_{obs}^{-1}(O) \cap \mathcal{L}(M)$". A major 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 [32]. If the answer is positive, one can build a process that monitors observable actions and raises an alarm when a fault has occurred.

The objective of *history diagnosis* is slightly different, and does not focus on faults. Starting from an observation O, one wants to know everything that may have occurred in the system while producing this observation. History diagnosis can be formalized as "given an observation O and a model M, compute the largest subset $X \subseteq \mathcal{L}(M)$ such that for every run $R \in X$ we have $\Pi_{obs}(R) = O$". In a context where observed events are reported in an asynchronous fashion to a diagnoser, an observation records only events that have reached the diagnoser. To handle this asynchrony, we can adapt our formalization of diagnosis and compute the largest set $X \subseteq \mathcal{L}(M)$ such that there exists O', an observation that contains O (this notion of containment might differ depending on the considered model but one can see O as a prefix of O') and such that for every $R \in X$ we have $\Pi_{obs}(R) = O'$. Note that the computed sublanguage X is not necessarily finite, nor regular. Whenever possible (this depends on the nature of O and M) we want to refine the history diagnosis as : "given an observation O and a model M, compute a model M' such that $\mathcal{L}(M') \subseteq \mathcal{L}(M)$, for every run $R \in \mathcal{L}(M')$, $\Pi_{obs}(R)$ is an observation that contains O, and such that $\mathcal{L}(M')$ is the largest language with such properties".

The *existence problem* is a simplified history diagnosis problem that consists in deciding whether some run of a model M can explain an observation O. This can be formalized as "given an observation O and a model M, does $\mathcal{L}(M) \cap \Pi_{obs}^{-1}(O) = \emptyset$ "?

To complete this taxonomy of diagnosis problems, let us add that fault or history diagnosis can be addressed offline (the observation is fixed, the system is stopped) or online (the diagnosis is built incrementally while the system is still running, and progresses with every event that is reported to the diagnoser and appended to the observation. In the rest of this paper, we will address the history diagnosis and existence problems in a setting where observations are partially ordered sets of events, collected asynchronously by a monitor. We will assume that each instrumented process provides a total ordering on the events it reports, and that some additional causal dependencies among events observed by distinct process may also be provided. This can be achieved if some processes tag their observation with additional information (for instance a vectorial clock). The model of the system is a High-level Message Sequence Chart. We will show that history diagnosis and existence from HMSCs are decidable problems, and can be addressed offline and online. The main advantages in using HMSCs are first the ability to perform diagnosis using models with infinite state space but finitely represented, and second avoiding an interleaved representation of runs of the model.

Let us now highlight the differences between diagnosis and *supervisory control*. Supervisory control consists in monitoring a system (often called the plant), and prevent the occurrence of some actions so that the system's behaviors remains within a predetermined specification. The problem addressed in *diagnosis* is slightly different. Supervisory control prevents some "bad actions" to occur. In a system equipped with a diagnoser, bad actions can still occur, but the diagnoser must raise an alarm when it has enough information showing that some undesired feature has occurred. The problem addressed by *history diagnosis* is not to raise alarms, but rather to provide explanations on how the system may have behaved from the observations collected so far. History diagnosis can be useful after the occurrence of a fault, to detect why the fault occurred. But it can also be used to obtain information on the behavior of a running system in non-faulty situations. History diagnosis returns all the possible explanations that are compatible with the observations regardless of whether they describe faulty behaviors. For fault diagnosis, when there exists observations for which some explanations contain a fault, and some others do not, the observed system is *not diagnosable* (one can not decide whether a fault has occurred or not). Such situation should be avoided. However, for history diagnosis, this situation is not a problem.

Let us briefly review some existing results in the domains of model-based fault diagnosis and history diagnosis. We do not claim exhaustiveness, but simply refer to approaches that are related to the work presented in this paper.

[32] proposes a fault diagnosis framework based on automata. The objective is to detect when a fault has occurred. Lafortune et al. show that in some cases, the system is not diagnosable, that is it may not be possible from an observation to claim whether a fault has occurred or not. [35] proposes a modular diagnosis solution with observers modeled as automata. The proposed approach allows the local diagnosers to communicate to build a more accurate view of what occurred within the system. The approach of [10] is close, but does not assume real-time communication among local diagnosers. [23] addresses offline and online diagnosis for Mealy machines, that is automata with input and outputs on transitions (the outputs of the machine are used to model the fact that an event is observed). The diagnosis consists in partitioning the state space of the system into subsets called *cells*, and deciding from observations in which cell the system is. Beyond the fact that these approaches detect occurrences of faults [32,35,10] or if the system has reached some particular sate, on may note that these approaches use finite state models.

[14] proposes an online modular diagnosis framework based on Petri nets. The local diagnosers and the system models are Petri nets. The local monitors are allowed to communicate to reach a global diagnosis, which is an estimation

of the current state of the system. The authors also show that with some good partitioning of the observed events, local diagnosers can achieve the same result as a global diagnoser. This work allows to deal with infinite state systems. One difference with our approach is that an exact model of the system is supposed to be known, allowing diagnosers to be defined as abstractions of the system's model. However, as already mentioned in this section, such a model is not always available.

[30] proposes history diagnosis from modular definition of a system using automata. Local diagnosis computed by each component are assembled to output a global diagnosis. [3] considers history diagnosis from a set of communicating automata with bound on communication buffers. This is essential, as unbounded communicating automata can simulate Turing machines, which makes history diagnosis an undecidable problem. So this restriction limits the expressive power of the model to that of finite automata. Hence, these approaches are modular, but address only diagnosis with finite-state models. This can be a drawback if the running system is not finite state, and exceeds for instance the bounds on communication channels imposed by the model.

[4] studies history diagnosis using safe Petri nets. Observations are defined as occurrence nets (acyclic Petri nets, that can be seen as partial orders). The approach uses an incremental construction of an unfolding net which embeds the observation. [8] studies history diagnosis using symbolic unfolding techniques from safe high-level parameterized Petri nets. The main objective of this work is to find a complete parameterized set of explanations. The approach builds a symbolic unfolding, that is an unfolding of the Petri net decorated with constraints on the parameters. This technique allows to separate the control flow and the parameterized values in the proposed explanations, but also to represent infinite sets of explanations. Similar techniques can be applied to time Petri nets [9]. The incremental aspect of these approaches is clearly well adapted for online diagnosis, but does not always allow for a finite representation of explanations. When unobservable events may have occurred an arbitrary number of times, unfolding may never stop, and the incremental approach fails. Note also that these three approaches deal with systems with a finite number of markings (infinity only comes from parameters in [8,9]).

Our work addresses history diagnosis for infinite state systems, and does not rely on unfolding techniques but rather on the construction of a model generating all (and possibly infinite sets of) explanations.

3 Scenarios

Scenarios are a popular formalism to define use cases of distributed systems. Several languages have been proposed [18,29], 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 [18]. 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 an 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 by process q of a message m sent by process p. 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.*
- *$\phi : E \longrightarrow \mathcal{P}$ 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) $(\mu \cup \bigcup_{p \in \mathcal{P}} \leq_p)^* \ = \ \leq$

Intuitively, the meaning of $e \leq f$ is that e must occur before f in any execution of the considered scenario, the meaning of $\phi(e) = p$ is that event e is executed by process p, and the meaning of $\mu(e) = f$ is that e is a message sending, and f is the corresponding reception. For a more detailed description of all MSC features, we refer interested readers to [18]. MSCs have a graphical representation which is close to their formal definition: the visual aspect of

[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.

an 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: every message sent in an MSC is also received within this MSC (and conversely). We will then say that MSCs are *communication closed*.

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

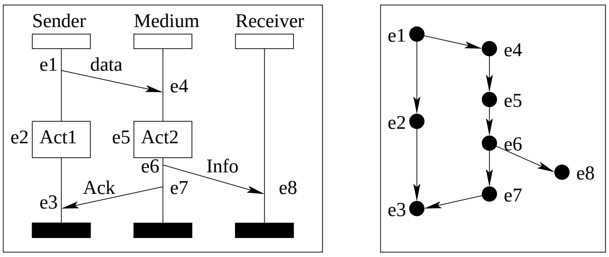

Fig. 2 An MSC Example and its underlying partial order

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* $\mu(e_1) = e_4$, $\mu(e_7) = e_3$.

$w = e_1 \dots e_{|E|}$ such that each event of E appears exactly once in w, and $\forall i, k \in 1..|E|$, $e_{i+k} \nleq e_i$. Back to the previous example, a possible linear extension of M is $e_1e_4e_2e_5e_6e_7e_8e_3$. A *linearization* of an MSC M is a word $\sigma = a_1 \dots . a_{|E|}$ 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 semantics of an MSC M is defined as the set of all its linearizations, and 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 a finite state machine 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 $x \in Dom(f_2)$.*

Intuitively, composing sequentially two MSCs M_1 and M_2 consists in drawing M_2 below M_1. The semantics of $M_1 \circ M_2$ is the set of linearizations $Lin(M_1 \circ M_2)$. Note that sequential composition does not impose synchronization among instances: in $M_1 \circ M_2$, one **needs not** wait for the execution of **all** events in M_1 before starting executing events in M_2. Indeed, events of M_1 and M_2 can be concurrent in $M_1 \circ M_2$ if they are located on distinct processes. If all events in M_1 and M_2 are located on distinct processes, then their sequential composition is simply the union of both models. However, for every common process p in M_1 and M_2, all event executed by p in M_1 must occur before events located on p in M_2 (this is the intuitive meaning of $\{(e_1, e_2) \in E_1 \times E_2 \mid \phi(e_1) = \phi(e_2)\}$ in the definition of $\leq_{1\circ 2}$).

Example 2 A sequential composition of two MSCs is shown in Figure 3. In the composition $M1 \circ M2$, action a and the sending of message m, for example, are still concurrent events. The linearization $P1!P2(m).P3!P2(n).P3(a).P2?P1(m).P2?P3(n)$ is a valid execution of $M_1 \circ M_2$. In particular, notice that executing $P3!P2(n)$ before $P2?P1(m)$ is allowed.

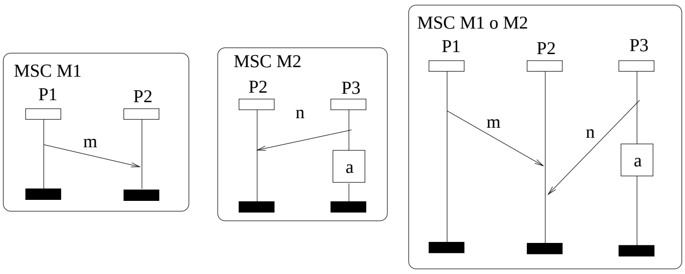

Fig. 3 Sequential composition of MSCs

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. An HMSC can be seen as a finite-state automaton labeled by MSCs, formally 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 an 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. An HMSC is then just the generator for a set of partial orders. We will call the *degree* of H the maximal number of transitions leaving any node of H.

Example 3 Consider the HMSC H in Figure 4. The initial node n_0 is depicted by the presence of a downward triangle, and the only final node n_1 is depicted by the presence of an upward triangle. The transitions of H are (n_0, M_1, n_0),(n_0, M_3, 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. The client can also send a notification message Alive to the server to keep its session alive (MSC $M3$). These two behaviors M_1 and M_3 can be iterated an arbitrary number of times (described by the loops 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 or Alive by the client.

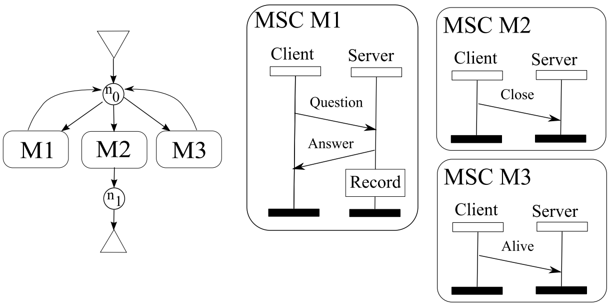

Fig. 4 High-level MSC

The whole terminology for automata applies to HMSCs. A *path* of H is a sequence of transitions $\rho = n_0 \xrightarrow{M_1} n_1 \dots \xrightarrow{M_k} n_k$ 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*. An 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. An HMSC can then be considered as the generator for the set of MSCs $\mathcal{F}_H = \{M_\rho \mid \rho \in \mathbb{P}_H\}$, and its semantics is given as the set of linearizations $\mathcal{L}_H = \bigcup\limits_{M \in \mathcal{F}_H} Lin(M)$.

Remark 1 Even if H is an automaton labeled by MSCs, its linearization language $\mathcal{L}_H$ may not be a regular language. For instance, the HMSC of Figure 4 does not have a regular set of behaviors (the Client process can send an arbitrary number of Alive messages before the first of them is received by the Server process). It has been shown that High-level Message Sequence Charts embed the expressive power of Mazurkiewicz traces [26], and rational relations. Consequently, several classical model-checking problems (language inclusion, equality, universality, vacuity of intersection,...) are undecidable for HMSCs. It is even undecidable in general to know whether the linearization language of an 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 finite-state machines, globally cooperative HMSCs [15] allow for decision procedures for inclusion, equality, and vacuity of intersection, etc. In the next sections, we show that history diagnosis with HMSCs as models is decidable without restriction. This means in particular that our results offer **a diagnosis solution for a class of infinite state models**, *which can be seen as an improvement with respect to solutions based on finite state models such as finite-state automata or safe Petri nets. Note also that increasing the expressive power of the system's model to embed the expressive power of Communicating Finite State Machines leads to undecidability.*

4 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 an MSC). A subset of events is monitored on several processes of the system: 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** of at most t_{obs} time units. The set of types of monitored events is defined as an observation alphabet Σ_{obs}. We hence consider that probes observe only events which type belongs to Σ_{obs}, and defining the contents of this alphabet is a mean to tune probes. We will also consider that for each process, the observation is a sequence, that is, the communication between local observers and the supervision architecture is FIFO. 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. 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 $(\mu_O \cup \bigcup_{p \in \mathcal{P}} \leq_{O_p})^* \subseteq \leq_O$.*

The events recorded in an observation are the information sent by probes: the fact that a message of type m was sent or received, that an internal action was executed, etc. We suppose that the logged events have the same form as the events in the HMSC that will be used to explain an observation, i.e. that no pre-processing of the logs is needed to compare it with our model of the system. From the definition, observations are a relaxed version of MSCs with less constraints on ordering: they are not necessarily communication-closed, as some message sendings of $E_{O_!}$ may not have an image through μ_O, and some receptions may not be the image of an emission a sending. 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. In an observation O, the intuitive meaning of $e \leq_O f$ is that **it was observed that** e is a causal predecessor of f. Note that $e \leq_O f$ necessarily means that e occurred before f (meaning that the date of occurrence of e is smaller than the date of occurrence of f), but that the converse is not true: an event e can occur before another event f, but yet the two events are not causally related. 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.

Example 4 We show an example of observation in Figure 5. 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 $x \leq_O z \leq_O y$ and $\nexists z \in E_O, x \leq_O z \leq_O y$. 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

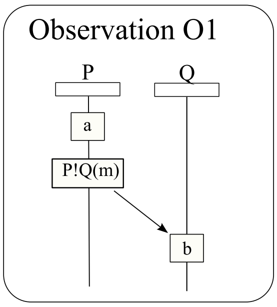

Fig. 5 An example observation

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 will say that a set of events $E \subseteq E_O$ is *causally closed in* 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 observations.

Definition 5 *Let $M = (E, \leq, \alpha, \mu, \phi)$ be an 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, ϕ) 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, an observation is a partial view of what occurred in a system, and the observed order among observed events might be less precise than the actual causal ordering of the real execution observed by the probes. 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, the systems described and their observation mechanisms are networks of machines communicating asynchronously. Events collected by probes on each site are sent asynchronously to the diagnoser. This means that when performing diagnosis, we have to take into account that some observed events were not yet received, and hence do not appear in the observation. 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 causally closed in 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 $\mu'_O \subseteq \mu_O$.*

Slightly abusing our definition, we will say that a set of events $E \subseteq E_O$ is a *prefix* of O if the restriction of O to E is a prefix of O. We will say that an MSC M and an observation O are consistent when O is an observation that might have been collected by probes during the execution of M. This can be formally 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 an 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 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}). We will also write $O \rhd M$ instead of $O \rhd_{\Sigma_{obs}} M$ when Σ_{obs} is clear from the context. 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 [27,25]. This matching is an embedding relation from an MSC M to another MSC N, that preserves events labeling and ordering. Such embedding means that N is a refinement of M. It was shown in [27] that this matching can be extended to HMSCs, and used for verification purposes. However, in this paper, we will only use matchings from finite observations to finite MSCs.

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 that $\alpha(e) \in \Sigma_{obs}$ and $\alpha(f) \in \Sigma_{obs}$. The fact that $h^{-1}_{O,M}(f)$ is defined implies that $h^{-1}_{O,M}(e)$ 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 and closed by precedence on each process). If $h_{O,M}$ does not preserve causal ordering for some events located on distinct processes, then O can not be a prefix of a sub-order of $\pi_{\Sigma_{obs}}(M)$. Conversely, if O is a prefix of a sub-order of $\pi_{\Sigma_{obs}}(M)$, then for every ordered pair of events $e \leq_O f$ in O, we have $h_{O,M}(e) \leq_M h_{O,M}(f)$. Similarly, for every $f \in h_{O,M}(E_O)$, all observable predecessors of f on the same process must be in $h_{O,M}(E_O)$, otherwise the observation O, which is totally ordered processwise, can not be a prefix of a sub-order of $\Pi_{\Sigma_{obs}}(M)$. □

Corollary 1 *Given an observation O such that $O \triangleright_{\Sigma_{obs}} M$ and an observation O' such that O' is a prefix of O, or O' is a sub-order of O, then $O' \triangleright_{\Sigma_{obs}} M$. Furthermore, if $O' = \Pi_{\Sigma'}(O)$ for some alphabet $\Sigma' \subseteq \Sigma_{obs}$, then $O' \triangleright_{\Sigma'} M$.*

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. □

Note that a prefix of a sub-order of an MSC M is not necessarily a sub-order of a prefix of M. Consider for instance the MSC M_1 in Figure 6. The observation that contains only one occurrence of a on P_1 and one occurrence of b on P_2 is a prefix of the sub-order of M_1 where all causal dependencies between the events on P_1 and the events on P_2 have been removed. However, this observation is not a suborder of any prefix of M_1. Indeed, any suborder of a prefix of M_1 that contains one occurrence of b must contain two occurrences of a.

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

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 $O_1 \triangleright_{\Sigma_{obs}} M_1$.*
- *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}. The event c in M_2 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. This example is interesting, as it illustrates the main difference between diagnosis from an interleaved and a non-interleaved representation. MSC M_3 indicates that events a and b are not causally related. Event a may occur before b, or the converse, or both event may occur concurrently. From the observation O_3 we learn that b could not occur before a. However, M_3 does not explain why b could not occur before a, and hence is not a valid explanation for O_3. In some sense, a non-interleaved model provides more information than an interleaved one (mainly on the causal dependencies among events), and this information can be exploited to improve accuracy of diagnosis.*
- *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*

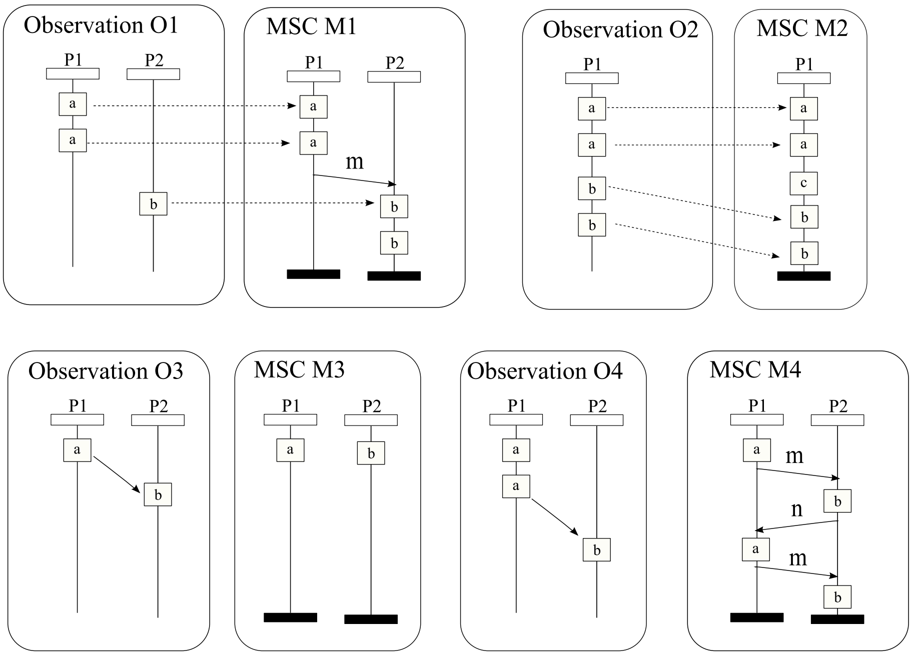

Fig. 6 Two matching examples w.r.t $\{a, b\}$ and two counter examples

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 an 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)).

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 $\mathbb{P}_{O,H} = \{\rho \in \mathbb{P}_H \mid O \triangleright_{\Sigma_{obs}} M_\rho\}$.*

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 5, 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 be embedded into an explanation.

Our goal is to build incrementally the set of all explanations provided by an 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 an HMSC. This leads us to introduce the notion of *compatibility* defined as follows:

Definition 9 *An MSC M is* compatible *with an observation O if and only if there exists an MSC M' such that $O \rhd_{\Sigma_{obs}} M \circ M'$. Given an 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 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 worth 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.

5 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. More formally, the problem can be stated as follows. Given an observation O of an instrumented system, and an HMSC H, build a new HMSC H' such that, for every accepting path ρ of H', we have $O \rhd M_\rho$

This generator H' can be defined as a product between the original HMSC and the observation, with synchronization on monitored events. In the rest of the paper, this new HMSC will be called a diagnosis HMSC (or simply a diagnosis). The nodes of a diagnosis HMSC are products 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 any node $n = (v, E_O)$ of a diagnosis HMSC, a path ρ leading to n should generate an MSC M_ρ that embeds O.

Next we outline some difficulties we will face in order to build a diagnosis HMSC. First, one can not decide if a path $\rho = n_0 \xrightarrow{M_1} n_1 \dots \xrightarrow{M_k} n_k$ is an explanation of O just by considering the projections $\Pi_{\Sigma_{obs}}(M_1), \dots, \Pi_{\Sigma_{obs}}(M_k)$. 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, a major 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. Intuitively, $g_{O,M}(p)$ recall the events of O preceding some event located on process p. More formally, for an observation O and an 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 $O_1 \circ O_2$ consist of 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.*

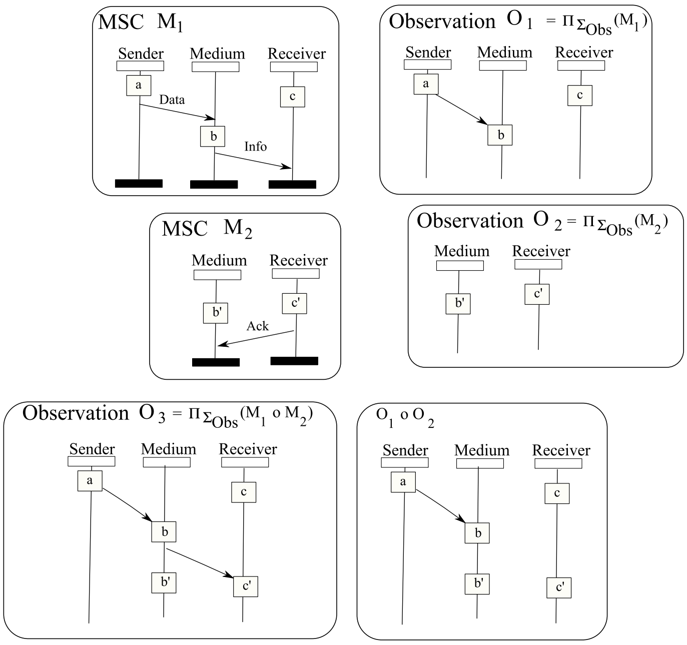

Fig. 7 Concatenation and Projection

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,*

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

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

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 an 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. □

We can now show how to compute the function $g_{O,M_1 \circ M_2}$ from g_{O,M_1} and M_2.

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 an 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 an 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 an HMSC H:

Definition 10 *Given an HMSC H and an observation O on an alphabet Σ_{obs}, the diagnosis HMSC $\mathcal{A}_{O,H}$ is defined as a tuple $\mathcal{A}_{O,H} = (Q, \delta, q_0, \mathcal{M}, F')$, 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 $E_{O p}$, or $E_{O p}$ 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 both $h(a), h(b)$ appear in MSCs preceding M (i.e. $a, b \in E$) and their ordering was already checked, or they are both events of M which are ordered in M, or $h(a)$ was observed before MSC M (i.e. $a \in E$), $h(b)$ is in M, and is a successor on the same process 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.

The diagnosis $\mathcal{A}_{O,H}$ is an HMSC with a particular shape: for every cycle c, all states in c have the same set of observed events and function g. Except for cycles over final nodes, transitions in cycles are labeled by MSCs that do not contain observable events. Hence, $\mathcal{A}_{O,H}$ can be seen as an acyclic graph connecting strongly connected components of H with unobservable behaviors. We will show later (in section 6.2) that this property can be used to reduce the size of the information kept in memory. The explanations provided by $\mathcal{A}_{O,H}$ of course include paths that go through unobservable components, however, we believe that the transitions on interest in $\mathcal{A}_{O,H}$ are the moves that make function g progress, i.e. that explain in which scenario an event appears, or why some causal dependencies among observed events hold.

Theorem 1 *[17] Let $\mathcal{A}_{O,H}$ be the diagnosis HMSC computed from O and H, and $\rho \in \mathbb{P}_H$. Then $O \triangleright_{\Sigma_{obs}} M_\rho$ 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 an MSC M_ρ such that $O \triangleright_{\Sigma_{obs}} M_\rho$, 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 $\rho \notin \mathbb{L}_{H,\mathcal{A}_{O,H}}$. If $\rho \notin \mathbb{P}_H$ then we are done. Now, let $\rho \in \mathbb{P}_H$. This path ρ is of the form $\rho = \rho_1\rho_2$ where ρ_1 is the longest sequence of transitions for which there exists a sequence of transitions $\rho' \in \mathbb{L}_{H,\mathcal{A}_{O,H}}$ such that $\rho' = \rho_1.\rho'_2$. 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 $i \in \mathcal{P}$ a prefix of O. 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.

Corollary 2 *Let H be an HMSC of degree d over p processes, labeled by MSCs of size at most m, and O be an observation. Then, computing $\mathcal{A}_{O,H}$ can be done in $O((m + p^2 + (m + p^2)^4) \times d \times |H| \times |O|^{|\mathcal{P}| \times |\mathcal{P}_{Obs}|})$.*

proof : Appending a transition, i.e. an MSC M to an existing $\mathcal{A}_{O,H}$ needs to verify that the matching relation computed so far can be extended. If $g(p)$ is recorded efficiently (for instance with a sorted list of events per process), then finding an extension of the embedding h from O to the currently built explanation consists in starting, for each process p with the successor of the last event seen so far, and comparing sequentially sequences of observable events on p in O and M. This can be done in linear time in the size of M. Now, each minimal event in $h^{-1}(E_M)$ can have at most p immediate predecessors. We can hence build a new order $P(M) = (E_M \cup Pred(M), \leq_{PM}, \alpha, \phi, \mu)$ in which $Pred(M)$ is the set of predecessors of $h^{-1}(E_M)$ in O, and $x \leq_{PM} y$ if $x \leq_M y$ or $x \in g(\phi(y))$. If the ordering in O is stored efficiently (for instance using immediate predecessor/successor lists), then building $P(M)$ can be done in $O(p^2)$. Intuitively, $P(M)$ represents the observable ordering according to what has been executed so far in an explanation and according to M. Now, it remains to show that the restriction of O to events in $h^{-1}(E_M \cup Pred(M))$ matches $P(M)$. Following Proposition 2, and as the matching relation needs not be recomputed, this can be done in $O((m+p^2)^4)$ (or even more efficiently in $O(4 \cdot (m+p^2)^2)$ if we consider the covering of order relations). For every state of $\mathcal{A}_{O,H}$, this test has to be performed at most d times.□

Note that even if the way observations and g are stored may influence the overall worst case time complexity, it remains a constant factor depending only on characteristics of the model H, and not on the size of the observation. Note also that complexity may increase with the size of the observation, but remains linear in the size of the model.

Remark 2 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 find only **minimal** paths, we can consider 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 5.2 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 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, $n_0 \xrightarrow{M1} n_0 \xrightarrow{M2}$

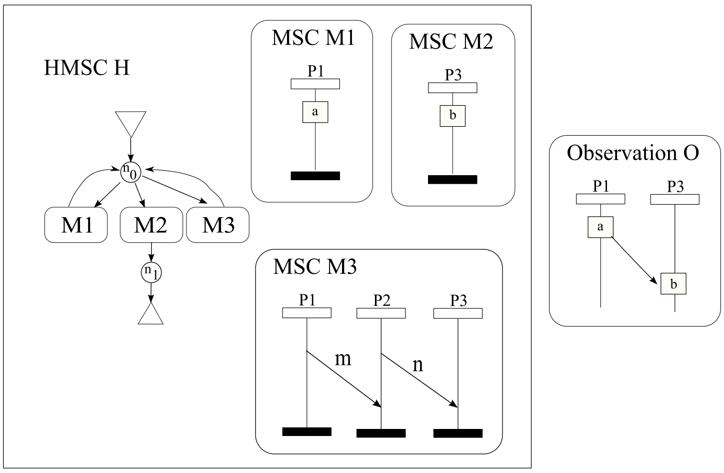

Fig. 8 An HMSC example and an observation

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 diagnosis $\mathcal{A}_{O,H}$ 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 HMSC. 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. To complete this example, let us focus on the construction of a particular transition from q_2 to q_3. State q_2 is of the form (n_0, E_2, g_2), where E_2 is the set containing a single event e_1 labeled by action a, located on process P_1, and $g_2(P1) = g_2(P2) = \{e_1\}$. We want to check if transition $(n_0, M3, n_0)$ of the HMSC model is compatible with what has been observed so far, and if so, compute a new state $q_3 = (n_0, E_3, g_3)$. As $M3$ does not contain observable events we necessarily have $E_3 = E_2$. However, as $e_1 \in g_2(P2)$, then in any path ρ considered so far in the HMSC that ends in configuration q_2, there exists a successor of $h(e_1)$ located on $P2$. Hence, after appending $(n_0, M3, n_0)$ to any path ending in q_2, we have $g_3(P1) = g_3(P2) = g_3(P3) = \{e_1\}$, as the message n from $P1$ to $P2$ in $M3$ creates a new event on $P3$ that is a successor of e_1.

5.1 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, $\exists ?\rho \in \mathbb{P}_H, O \rhd M_\rho$. This is equivalent to answering the question : "does $\mathbb{L}_{H,\mathcal{A}_{O,H}} = \emptyset$? "

Theorem 2 *Let H be an 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 $O \rhd_{\Sigma_{obs}} M_\rho$. Let us suppose that ρ is a path of length greater than $|O|^2 \times |\mathcal{P}| \times |H|$. 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 $|O| \times |\mathcal{P}| \times |H|$ 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 $|O| \times |\mathcal{P}| \times |H|$, it necessarily contains

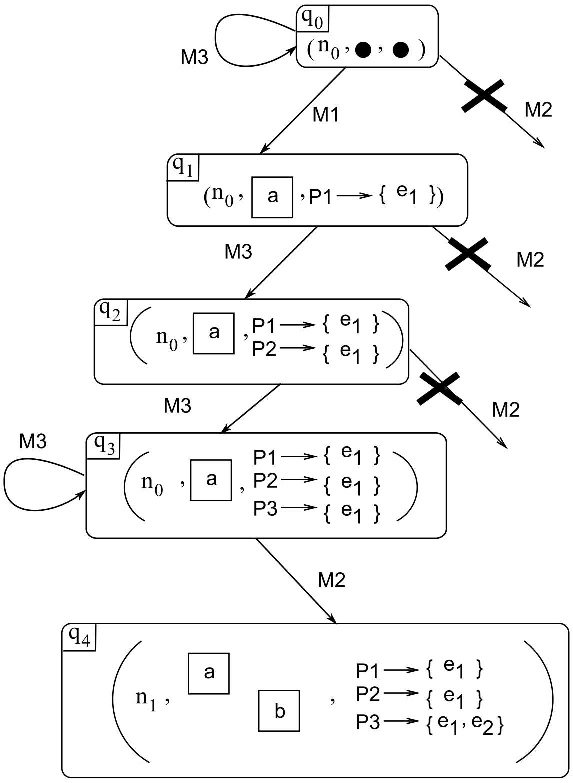

Fig. 9 A diagnosis HMSC

at least $|O| \times |\mathcal{P}|$ cycles of H, and hence it is a sequence of transitions of the form $\rho' = u_1.\beta_1.u_2.\beta_2 \dots \beta_{|O|\times|\mathcal{P}|}.u_{|O|\times|\mathcal{P}|+1}$, 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 $\rho_1.u_1\beta_1 \dots \beta_{i-1}.u_i.u_{i+1}.\beta_{i+1} \dots u_{|O|\times|\mathcal{P}|+1}.\rho_2$. 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 $i \in 1..|O|\times|\mathcal{P}|-1$, 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 $g_{|O|\times|\mathcal{P}|-1}(p) = E$ for every p, and $\beta_{|O|\times|\mathcal{P}|}$ is mapped to a loop of $\mathcal{A}_{O,H}$, and can be removed from ρ to obtain a smaller explanation $\rho_1.u_1\beta_1 \dots \beta_{|O|\times|\mathcal{P}|}.u_{|O|\times|\mathcal{P}|+1}.\rho_2$. Hence, if an explanation exists for an observation O, then there is necessarily an explanation of length at most $|O|^2 \times |\mathcal{P}| \times |H|$. We can then select in polynomial time a path of H of length lower than $|O|^2 \times |\mathcal{P}| \times |H|$.

Now let us show that for such 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 $M = (E, \leq, \alpha, \mu, \phi)$ 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 $O(n_1 + n_2 + m_1 + m_2 + p \times (n_1 \times m_1 + n_2 \times m_2 + m_1 \times m_2))$ for concatenation. Computing M_ρ resumes to k concatenations of MSCs with less than $k \times n$ events and a causal ordering relation of size smaller than $(k \times n)^2$, and can hence be performed in polynomial time, and results in an MSC with at most $k \times n$ events and a causal ordering relation of size at most $(k \times n)^2$. Then, from proposition 2, verifying that $O \rhd_{\Sigma_{obs}} M_\rho$ can be done at most in $O(k \times n + | \leq_O | \times (k \times n)^2)$. 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 $v_1, \dots, v_n$, 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 $v_1, \dots, v_n$ 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 an 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 $\mathcal{P} = \{P_{v_1}, \dots, P_{v_n}\} \cup \{P_{c_1}, \dots, P_{c_n}\}$. The observation alphabet Σ_{obs} is composed of $m+1$ letters $\{a_0, a_{c_1}, \dots, a_{c_m}\}$. The observation O contains 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 $Q = \{q_0, q_e, q_f\} \cup \{q_{v_1}, \dots, q_{v_n}\}$, and is labeled by MSCs $Start, End$ and $T_{v_1}, \dots, T_{v_n}, F_{v_1}, \dots, F_{v_n}$. 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 $i \in 1..n-1$, and two transitions from q_{v_n} to q_e respectively labeled by T_n and F_n. An occurrence of each action in Σ_{obs} 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}. □

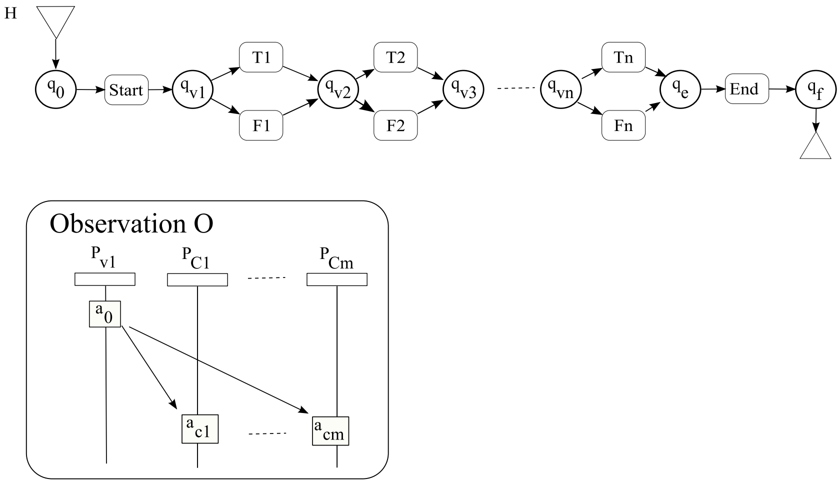

Fig. 10 Encoding SAT problems with an existence problem

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 *[17] Let H be an 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 a diagnosis HMSC $\mathcal{A}_{O,H}$ with observation alphabet Σ_{obs} such that $\mathbb{L}_{\mathcal{A}_{O,H}} \neq \emptyset$. 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 $\Pi_{\Sigma'_{obs} \cap \Sigma_p}(M_\rho)$, and then M_ρ does not embed O with observation alphabet Σ_{obs}, or there exists 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_ρ can not embed O. Hence we can not have $\mathbb{L}_{\mathcal{A}_{O,H}} \neq \emptyset$. □

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.

5.2 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, the centralized diagnosis amounts to build a diagnosis HMSC of exponential size in the number of considered processes. We can also 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, as function g memorizes prefixes of O. This situation may occur when the HMSC model H is labeled by MSCs which contain at most one observable event per transition of H. Despite this fact, representing observation as a partially ordered set of events still makes sense. It allows to differentiate between events that are concurrent and ordered, which interleaved models can not do. Furthermore, linearizing an observation a priori imposes a systematic exponential blowup, and forces representing linearizations that will never be considered otherwise by the diagnosis during the construction of $\mathcal{A}_{O,H}$. Hence in practice, the compact partial order representation for O can only be more efficient than an interleaved counterpart.

A solution to reduce complexity is to split the diagnosis computation into several simpler sub-problems that can be easily distributed to solve the diagnosis concurrently. Formally, given an HMSC model H and an observation O, compute $\mathcal{A}_{O,H}$ as a product of smaller problems $\mathcal{A}_{O,H} = \mathcal{A}_{O_1,H} \otimes \dots \mathcal{A}_{O_1,H}$.

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. Splitting the diagnosis into several smaller sub-problems, which solutions can be computed **independently** by distinct machines, and then combining local solutions allows to produce a global answer. We will also show at the end of this section that properties of splitting also open the way for distributed diagnosis frameworks.

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 diagnosis HMSC $\mathcal{A}_{p,q} = \mathcal{A}_{\pi_{\Sigma'}(O),H}$ 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 HMSCs.

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

We can now formalize the distribution of diagnosis: given an observation O and an HMSC H, project O on pairs of processes to obtain smaller observations $O_1, \dots O_k$, compute all $\mathcal{A}_{O_i,H}$ for $i \in 1..k$, and then compute $\mathcal{A}_{O_1,H} \otimes \cdots \otimes \mathcal{A}_{O_k,H}$. Theorem 3 below shows that the product is a diagnosis automaton for O, as 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*
$\mathbb{L}_{H,\mathcal{A}_{O,H}} = \mathbb{L}_{H,\mathcal{A}_{\otimes}}$, *where* $\mathcal{A}_{\otimes} = \bigotimes_{p \neq q \in \mathcal{P}_{Obs}} \mathcal{A}_{p,q}$.

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 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 $\mathbb{L}_{H,\mathcal{A}_{O,H}} = \mathbb{P}_{O,H} \subseteq \mathbb{L}_{H,\mathcal{A}_{\otimes}}$.

For the second inclusion, let $\rho \in \mathbb{L}_{H,\mathcal{A}_{\otimes}}$. 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 $h_{O,M_\rho}(e) = h_{p,q}(e) \leq h_{p,q}(f) = h_{O,M_\rho}(f)$.

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 $O(|O|^{2|\mathcal{P}|} \times |H|)$. 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 $\frac{|\mathcal{P}_{Obs}| \times (|\mathcal{P}_{Obs}|-1)}{2}$ 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 $\frac{|\mathcal{P}_{Obs}| \times (|\mathcal{P}_{Obs}|-1)}{2}$ diagnosis if none of the local results is empty. Note that the size of a local diagnosis is not necessarily smaller than the size of the original diagnosis HMSC 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.

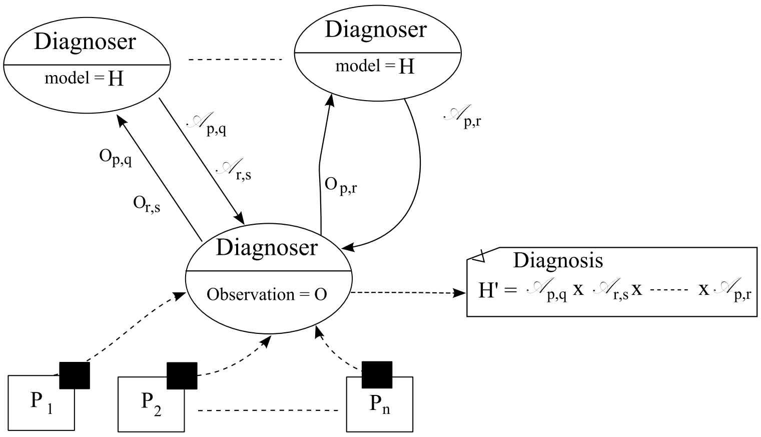

Fig. 11 Decentralized Diagnosis/Existence problem

This allows for a decentralized version of our initial diagnosis architecture, depicted in Figure 11. There is still a central diagnoser that collects the observation. Its role is then to compute a projection $\Pi_{\Sigma_{obs} \cap (\Sigma_p \cup \Sigma_q)}(O)$ for every pair $p \neq q$ considered, and send it to local diagnosers. Each local diagnoser possesses 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 received diagnosis 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 $\frac{|\mathcal{P}_{Obs}|\times(|\mathcal{P}_{Obs}|-1)}{2}$ local diagnosers, as each of them can compute more than one local diagnosis.

Theorem 3 shows 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. Formalizing the problem, we want to split the boolean existence problem EP: "is $\mathbb{L}_{H,\mathcal{A}_{O,H}}$ empty" into several boolean problems $EP_{p,q}$: "is $\mathbb{L}_{H,\mathcal{A}_{p,q}}$ empty" for p, q ranging over pairs of distinct observable processes. The answer to each local problem $EP_{p,q}$ is either true ($\mathbb{L}_{H,\mathcal{A}_{p,q}}$ is empty) or false ($\mathbb{L}_{H,\mathcal{A}_{p,q}}$ is not empty).

For the existence problem, we know that as soon as a local diagnosis is empty, H provides no explanation for O. Hence, if the disjunction $\bigvee EP_{p,q}$ is true, the answer to the global existence problem is also true (disjunction yields no wrong positives). However, disjunction of local answers may yield wrong negatives. Indeed, there are cases where all local diagnosis HMSC have a 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 a HMSC with empty language (i.e. $\mathbb{L}_{H,\mathcal{A}_{O,H}} = \bigcap_{p,q\in\mathcal{P}^2_{Obs}} \mathbb{L}_{H,\mathcal{A}_{p,q}} = \emptyset$). The example of Figure 12 shows such case.

To avoid wrong negatives, one must compute a product $\mathcal{A}_{\bigotimes} = \mathcal{A}_{p_1,q_1} \otimes \cdots \otimes \mathcal{A}_{p_n,q_n}$ of local diagnosis obtained for each pair of distinct processes in $\mathcal{P}_{obs}$. Then the answer to the global problem EP is the answer to the question "is $\mathbb{L}_{H,\mathcal{A}_{\bigotimes}}$ empty"', and may differ from $\bigvee EP_{p,q}$. This product can always be built (it is a simple product of HMSCs, as proposed in definition 11). 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.

An immediate question is how to ensure that existence can be addressed as a disjunction of smaller existence problems, i.e., $EP = \bigvee EP_{p,q}$? This property holds only if for distinct pairs $\{p, q\}$ and $\{p', q'\}$ of observable processes such that $\mathbb{L}_{H,\mathcal{A}_{p,q}} \neq \emptyset$ and $\mathbb{L}_{H,\mathcal{A}_{p',q'}} \neq \emptyset$ the intersection $\mathbb{L}_{H,\mathcal{A}_{p,q}} \cap \mathbb{L}_{H,\mathcal{A}_{p',q'}}$ is also non-empty. It means that for every observation O, all $\mathcal{A}_{p,q}$ with non-empty language must follow at least a common path. This property seems difficult to ensure, except when the running system and the model used to perform diagnosis it have the same behaviors, and that the existence problem can be trivially answered without computing the product: an observation always has an explanation, so $\mathbb{L}_{H,\mathcal{A}_{O,H}} \neq \emptyset$.

Example 8 Consider the HMSC H and the observation O of figure 12. For each pair of processes $p \neq q$ *in* $\{P1, P2, P3\} \times \{P1, P2, P3\}$, *it is possible to*

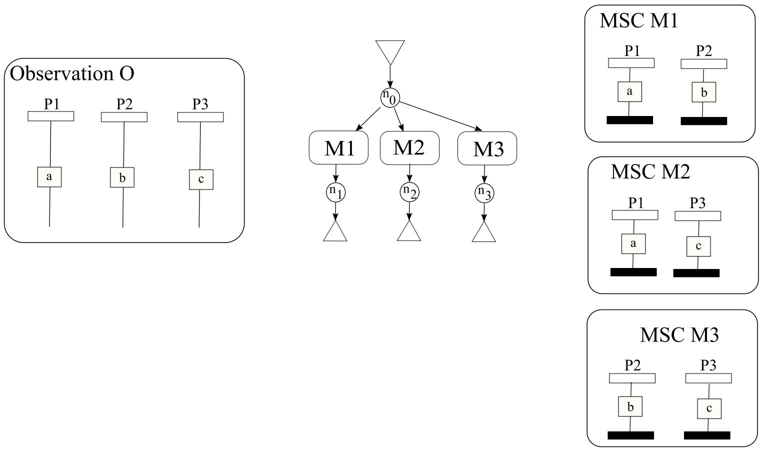

Fig. 12 Example showing that computing a product is needed for decentralized existence

find an explanation for $\Pi_{\Sigma_{obs}\cap(\Sigma_p\cup\Sigma_q)}(O)$*. However,* H *does not contain an 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 diagnosis HMSC $\mathcal{A}_p = \mathcal{A}_{\pi_{\Sigma_{obs}\cap\Sigma_p}(O),H}$ 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 HMSCs 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 pairs $p \neq q \in \mathcal{P}_{obs}$ for which there exists 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 opens the way for a distributed diagnosis framework. Indeed, one can easily adapt Theorem 3 to show that if $\mathcal{P}$ can be partitionned into clusters of processes $\mathcal{P}_1, \ldots, \mathcal{P}_k$ in such a way that for every observation O, and for every pair $e, f \in E_O$ of events, $e \leq_O f$ implies that e and f are located in the same cluster, then one can compute a global diagnosis for each $\mathcal{P}_i, i \in 1..k$, and build a global diagnosis as a product of local solutions. Hence, one can design local diagnosers, and combine their solutions when needed. If clusters are small, the construction of local diagnosis is of reasonable complexity (but computing the whole product may still be costly).

6 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 problems online, that is compute solutions incrementally as observed events arrive from probes. A naïve approach to solve these problems online with the framework described in section 5 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:

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

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 $n+1$. Formally, given an observation O, an HMSC H, a diagnosis $\mathcal{A}_{O,H}$ and an event e, the online diagnosis problem consists in computing a new diagnosis HMSC H' such that $O.e \triangleright M_\rho$ for every final path of H' as a function $H' = IncrementDiagnosis(\mathcal{A}_{O,H}, O, e)$. 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 built so far needs to be remembered to continue diagnosis on the fly. We address the following two 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 diagnosis.

Before entering into the technical details let us describe how new occurrences of observed events are collected and assembled to produce a coherent observation. Consider again Figure 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 receives newly observed events from probes, and updates observation and diagnosis accordingly. The only assumptions on communications from probes to the diagnoser is that the communication channels are FIFO and asynchronous. Furthermore, two events observed on the same machine are necessarily ordered. If two events e, f have been observed on distinct processes and e precedes f in the execution, f might be received by the diagnoser **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. This tagging mechanism is ensured by probes: in the simplest setting, probes just attach an increasing sequence number to their observations, in a more elaborated settings, they can maintain vectorial clocks. Let us call O the observation that has 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 do 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, all its predecessors in O are known, and have also been appended. We will also denote by $O.e$ the observation obtained by appending an event e to O. The rest of this section is organized as follows: Section 6.1 shows that some parts of a diagnosis do not influence the online construction of a diagnosis. Section 6.2 shows that one can compute a small structure summarizing a complete diagnosis to save space. Section 6.3 shows how to abstract the useless information identified in section 6.1 to save space. Section 6.4 studies the necessary conditions under which online diagnosis runs with finite memory. Section 6.5 builds on the results of former sections to propose an online existence algorithm, and section 6.6 summarizes the results of the whole section 6.

6.1 Safe parts of a diagnosis

In this section, we first define formally the parts 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 $\mathcal{A}_{O,H} = (Q, \delta, q_0, \mathcal{M}, F)$ be the diagnosis HMSC computed from observation O and HMSC H. Let $t = (q_{k-1} \xrightarrow{M_k} q_k)$ be a transition of $\mathcal{A}_{O,H}$. We will say that t is* safe *if* every observable event of M_k has been observed: *for every 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). Let T be a set of transitions of $\mathcal{A}_{O,H}$. We say that T is a* safe part *of $\mathcal{A}_{O,H}$ if it contains only safe transitions.*

Note that two paths ρ_1, ρ_2 of $\mathcal{A}_{O,H}$ going through the same transition $t = \Big((n, E, g) \xrightarrow{M} (n, E', g')\Big)$ have the same impact on t 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 mapping e and f consists in checking that they have identical labels, and are both the k^{th} observed event on their process. This does not depend on the path followed, 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* $T_{\max}$. For representation reasons, we will be interested only in *maximal connected safe parts*, that is the connected components of $T_{\max}$. We will say that two transitions $q_1 \xrightarrow{M_1} q_1'$ and $q_2 \xrightarrow{M_2} q_2'$ are *connected* if $q_1 = q_2'$ or $q_2 = q_1'$.

It is easy to build $T_{\max}$ in linear time in the number of transitions of $\mathcal{A}_{o,H}$: $(n, E, g) \xrightarrow{M} (n, E', g') \in T_{\max}$ iff $|E'| - |E| = |\pi_{\Sigma_{Obs}}(M)|$ (there are as many events in $E' \setminus E$ as observable events in M). One can also compute every connected component of $T_{\max}$ in linear time in $|T_{\max}|$. In the rest of the paper, we will say that an event e in an 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 $h_{O,M_\rho}(f) = e$).

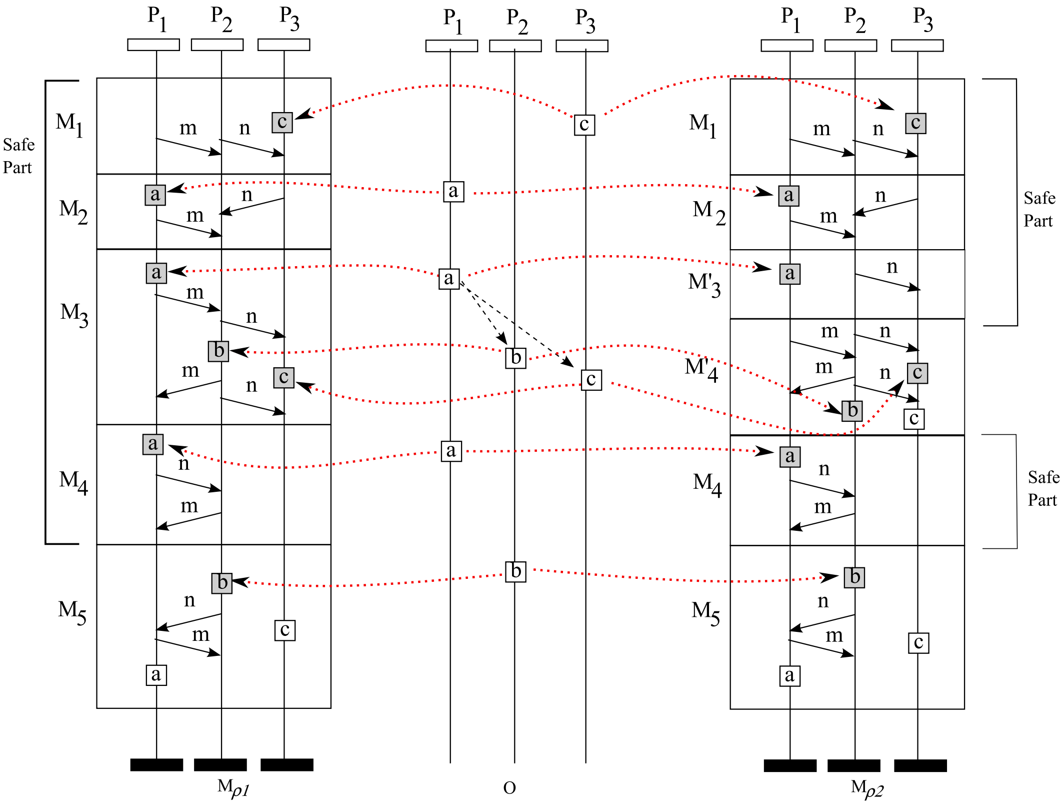

Fig. 13 Observations and safe parts

Example 9 Let us consider Figure 13, which shows an observation (in the center of the figure) and two paths of a diagnosis $\mathcal{A}_{O,H}$, $\rho_1 = q_0 \xrightarrow{M_1} q_1 \xrightarrow{M_2}$

$q_2 \xrightarrow{M_3} q_3 \xrightarrow{M_4} q_4 \xrightarrow{M_5} q_5$ and $\rho_2 = q_0 \xrightarrow{M_1} q_1 \xrightarrow{M_2} q_2 \xrightarrow{M'_3} q'_3 \xrightarrow{M'_4} q_3 \xrightarrow{M_4} q_4 \xrightarrow{M_5} q_5$. The observation alphabet is $\Sigma_{obs} = \{P_1(a), P_2(b), P_3(c)\}$. Paths 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 $T_1 = \{q_0 \xrightarrow{M_1} q_1 \xrightarrow{M_2} q_2 \xrightarrow{M_3} q_3 \xrightarrow{M_4} q_4\}$. The maximal safe part of ρ_2 is $T_2 = \{q_0 \xrightarrow{M_1} q_1 \xrightarrow{M_2} q_2 \xrightarrow{M'_3} q'_3\} \cup \{q_3 \xrightarrow{M_4} q_4\}$. 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. If $\mathcal{A}_{O,H}$ contains only paths ρ_1 and ρ_2 then we have $T_{max} = T_1 \cup T_2$.

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. Nevertheless, 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 $\rho = t_1.t_2 \dots t_n$ 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 $\pi_{\Sigma_{Obs}}(M) \setminus h_{O,M_\rho}(E' \setminus E)$ 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 $f' \leq_M f$, and either $e' \in g(\phi(f'))$, or $h_{O,M_\rho}(f') = e'$. Intuitively, this means that all the images of predecessors of e are also predecessors of the image of e in the explanations built.*
- *For every transition $t_i = (n_i, E_i, g_i) \xrightarrow{M_i} (n'_i, E'_i, g'_i)$ in $t_1 \dots t_{k-1}$, $\pi_{\Sigma_O, \phi(e)}(M) = \pi_{\Sigma_{Obs}, \phi(e)}(E' \setminus E)$, where $\pi_{\Sigma_{Obs}, p}(X)$ denotes the restriction of X to events located on process p and with labels in Σ_{Obs}. Intuitively, this last property means that M is the first MSC in path ρ that contains an unsafe event on process $\phi(e)$.*

Proof: This can easily be shown by contradiction. □

Notice that since $|E'| - |E| \neq |\pi_{\Sigma_{Obs}}(M)|$, $t_k \notin T_{\max}$. Proposition 6 shows how to ensure that an embedding exists from observation $O.e$ to the MSC M_ρ labeling a path ρ. This property should be checked when event e arrives at the diagnoser. Proposition 6 also indicates that when diagnosis is built incrementally, $T_{\max}$ does not influence the next steps of the construction (this holds even for connected component of $T_{\max}$ that do not contain the initial state). Hence, we can just forget (or store somewhere for later use) $T_{\max}$, 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 HMSC, 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.

Proposition 6 only shows how to verify that observation $O.e$ is still embedded in a path of the diagnosis built so far for observation O. Now, as new events arrive, the observation and the diagnosis HMSC should be updated jointly. The following propositions show how to update $\mathcal{A}_{O,H}$ to $\mathcal{A}_{O.e,H}$. We first show how to update a single path, then extend this construction to a complete diagnosis HMSC.

Proposition 7 *Let $\rho = t_1 \dots t_k$ 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 $\mathbb{L}_{\mathcal{A}_{O',H}}$.*

Proof: This can easily be shown by contradiction. □

This proposition shows how to update a path (and hence a diagnosis HMSC) 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 $h_{O.e,M_\rho}(e) \leq f$ 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 has to append new transitions to this path to provide explanations for $O.e$. This is shown in the following proposition.

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, and*
- *for each $e' <_{O'} e$, there exists $f' \in M_\gamma$, $f' \leq f$, and $e' \in g(\phi(f'))$.*

Proof: This can easily be shown by contradiction. □

Proposition 8 shows how a path of $\mathcal{A}_{O,H}$ can be extended to explain a new observation $O.e$. From propositions 7 and 8, an algorithm to build a diagnosis incrementally looks straightforward: when a new event e arrives, find the minimal transitions that are not in T_{max} containing an event explaining e. Find the maximal transitions for which all observable events on $\phi(e)$ are the image of some event in O, and extend them to find an explanation for e. However, there may exist an infinite number of possible extensions for maximal transitions. The next section shows that it is sufficient to consider extensions of diagnosis with acyclic paths, and to work with a reduced version of the diagnosis HMSC.

6.2 Summarizing a diagnosis

As already mentioned in section 5 (remark 2), the diagnosis $\mathcal{A}_{O,H}$ built from an HMSC H and an observation O does not provide the smallest paths leading to an explanation. First, the construction of $\mathcal{A}_{O,H}$ may 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 one can remember a smaller structure called a *summary* during online construction and then 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 HMSC $\mathcal{A}_{O,H}$ 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 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. This restriction works because embeddings are unique relations: if one can embed O into M_ρ, that is build a function h_{O,M_ρ}, then h_{O,M_ρ} is also the embedding function from O to any MSC $M_{\rho.\rho'}$ obtained with an extension of ρ.

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 $O(|\mathcal{P}| \times |H| \times 2^{|\mathcal{P}|})$.*

Proof: We have to search an event f such that $\alpha(f) = \alpha(e)$ in all acyclic paths 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 $h_{O,M\rho}$ will be predecessors of f. For this, we can build a finite-state 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 $|\mathcal{P}| \times |H| \times 2^{|\mathcal{P}|}$ such states to explore to find minimal extensions explaining $O.e$. □

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 used in this proof is not a set of paths, but a finite-state automaton. In the sequel, we will denote by $Ext(q, e)$ the finite-state automaton built from a state q of $\mathcal{A}_{O,H}$ to find a minimal and acyclic explanation containing a new event e. We can now use this definition to refine the online construction of diagnosis. When a new event e arrives, find the minimal transitions that are not in T_{max} containing an event explaining e, and update $\mathcal{A}_{O,H}$ accordingly (as defined in proposition 7). For every maximal transition $t = (q, M, q')$ for which all observable events on $\phi(e)$ are the image of some event in O, compute

$Ext(q', e)$, and append it to $\mathcal{A}_{O,H}$. The obtained result is not yet a diagnosis, as it does not contain all paths embedding $O.e$. However, we can show that a diagnosis can still be constructed from this smaller structure.

The next definition and proposition show that the diagnosis HMSC built in offline diagnosis in section 5 is an acyclic HMSC 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 from this acyclic diagnosis.

Definition 15 *Let $\mathcal{A}_{O,H} = (Q, \delta, \mathcal{M}, q_0, F)$ be the diagnosis computed from observation O and HMSC H. A* silent transition *of $\mathcal{A}_{O,H}$ is a transition of the form $(n, E, g) \xrightarrow{M} (n', E, G)$, i.e. that does not change the set of observed events or the value of $g(p)$ for every $p \in \mathcal{P}$. The* summary *of $\mathcal{A}_{O,H}$ is the diagnosis HMSC $\overline{\mathcal{A}_{O,H}}$ obtained by:*

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

Note that the first step of summary construction does not remove all silent transitions. Some silent transitions are needed to guarantee that transitions with observable events remain accessible. Hence, only silent cycles, which obviously do not change the set of observed events, function g, or connections among nodes of $\mathcal{A}_{O,H}$ 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 $\mathcal{A}_{O,H}$. The HMSC $\overline{\mathcal{A}_{O,H}}$ is exactly the structure that is built incrementally by considering acyclic extensions of maximal transitions in $\mathcal{A}_{O,H}$ when new events are observed.

Example 11 Consider the example of Figure 9. Obtaining a summary from this diagnosis HMSC 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 HMSC 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. Each transition in $\overline{\mathcal{A}_{O,H}}$ is a move towards a non-silent transition, and when a non silent transition is taken 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 a diagnosis HMSC, 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 HMSC. 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 diagnosis $\mathcal{A}_{O,H}$ which can be of size up to $(|H| \times |\overline{\mathcal{A}_{O,H}}|)$.

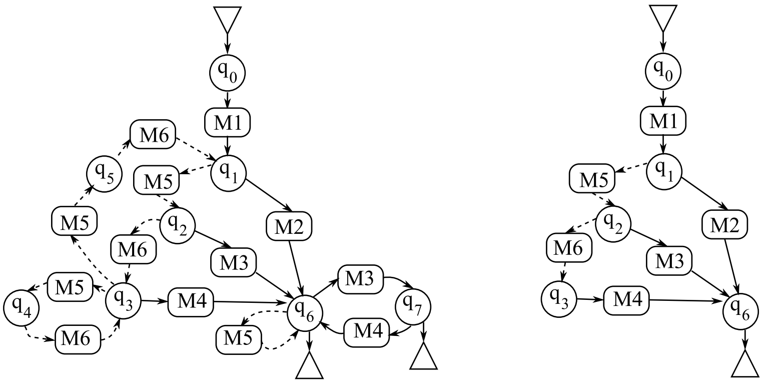

Fig. 14 A diagnosis and its summary

Example 12 Consider the diagnosis HMSC in the left part of Figure 14. Transitions labeled by silent MSCs are symbolized by dashed arrows. Note on this example that all successors of an accepting state are also accepting. Note also that cycles over states that are not accepting are exclusively composed of silent transitions. The diagnosis HMSC at the right of the figure is the summary of the diagnosis HMSC 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 HMSC to coaccessible nodes. 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* M_2, M_3, M_4.

6.3 Abstraction of a diagnosis

So far, we have seen that it is sufficient to build a summary online to obtain a complete diagnosis HMSC generating all explanations of an observation. We have also seen that when a new event is observed, some transitions of a diagnosis HMSC called the safe part, do not influence the existence of an embedding. Hence, this safe part 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.

Note that if the objective is to perform history diagnosis, safe parts **must** be stored in order to output a correct result. If the objective is only existence checking, then storing or not safe parts does not modify the result returned by the existence algorithm presented in section 6.5.

Definition 16 *Let $\mathcal{A}_{O,H}$ be a diagnosis HMSC let $T_{\max}$ be the set of safe transitions of $\mathcal{A}_{O,H}$, and let $F \subseteq T_{\max}$ be a maximal connected subset of $T_{\max}$. 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 describes whether in a set of transitions F, one can go from a state q to a state q' using only transitions of F. It 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 q of In_F, ρ' starts in a state q' of Out_F, and $(q, q') \in T(Mod_F)$ then an observation $O' \sqsupseteq O$ has an embedding in $\rho.\rho_F.\rho'$ for every ρ_F path of F from q to q' if and only if $O' \setminus E_{\rho_F}$ and $\rho.\rho'$ satisfy the conditions of proposition 6.

Hence, a diagnosis HMSC 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 $T_{\max}$).

First, we determine $T_{\max}$ as shown in section 6.1. 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 $T_{\max}$, we create a new module Mod_F in $\mathcal{A}_{O,H}$, summarized as: for each state q of $\mathcal{A}_{O,H}$ such that there exists an outgoing transition in F and an ingoing transition not in F, q 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, q 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 q and out-state q' of Mod_F such that there exists a path with transitions of F from q to q', we create a module-transition from q to q'. 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 unsafe transitions and modules following propositions 7 and 9. This structure with modules can be used exactly as $\mathcal{A}_{O,H}$, that is we can update its transitions, or extend it with new paths. The actions to perform to update $\mathcal{A}_{O,H}$ will depend on the nature of transitions in the unsafe part. 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 observable event on process $\phi(e)$ in this path and it is compatible with e). Slightly abusing the notation, we write $h(E)$ instead of writing $h_{O,M_\rho}(E)$ for each path ρ that ends at transition t. Note that conditions on $h_{O,M_\rho}(E)$ can easily be checked using the information available in state (n, E, g), 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 $f' \leq f$, and either $e' \in g(\phi(f'))$ or $h(f) = e'$ (the extension of the mapping with the pair (e, f) preserves observed causal ordering).

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 $f' \leq f$, we have $e' \notin g(\phi(f'))$ and $f' \neq h(e')$. (there exists no predecessor of f explaining the fact that e' causally precedes e)

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 $|h(E_{\phi(e)})| = |\{f' \in E_M \mid \alpha(f) \in \Sigma_O \wedge f' <_\phi (e) f\}|$
   – t has a successor transition t' in $\mathcal{A}_{O,H}$.

e) t is a maximal transition in $\mathcal{A}_{O,H}$ for process $\phi(e)$, i.e. it has no successor transition in $\mathcal{A}_{O,H}$, 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** a minimal transition containing unobserved events on process $\phi(e)$ along any path containing this transition, i.e. there exists $t' = q \xrightarrow{M'} q'$ which satisfies condition *a*), and such that there exists a path from q' to t in $\mathcal{A}_{O,H}$

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 transitions ending paths which embed $O \cdot e$, without adding new transitions to $\mathcal{A}_{O,H}$. Cases $b), c$) correspond to transitions ending paths which do not embed $O \cdot e$ (because the type of the first observable event on process $\phi(e)$ differs from the type of e (case b), or because some causality $e' \leq e$ of O is not embedded (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. However, paths ending with this transition have already been extended, as there exists a successor transition. Such transition need not be considered to explain e.

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. We have to extend $\mathcal{A}_{O,H}$ from the states reached by such transitions to find an explanation for e.

Case f) corresponds to transitions for which there exists a preceding transition still containing an unobserved event on $\phi(e)$, hence no event of M should be mapped to e. However, these transitions can be used later to explain further events.

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 HMSC. 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 HMSC.

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 algorithm 1.
For the sake of readability, we have split this algorithm into several phases that classify transitions, update them, and extend 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 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 $O(|\mathcal{A}_{O,H}| + Ke \times |\mathcal{P}| \times |H| \times 2^{|\mathcal{P}|})$, where Ke is the number of type e transitions, and can be computed in $O(|\mathcal{A}_{O,H}| + Ke \times |\mathcal{P}| \times |H| \times 2^{|\mathcal{P}|})$.*

proof : Finding $T_{\max}$ 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 $T_{\max}$, T_b and T_c can be done at the

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

```
INPUT: An HMSC $H$, a summary $\mathcal{A}_{O,H}$, an observation $O$, a newly observed even $e$
OUTPUT: a new summary $\mathcal{A}_{O\circ\{e\},H}$
Compute $T_{\max}$
/* nb : $T_{\max}$ may contain module transitions */
Compute $C$, connected components of $T_{\max}$
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 $\mathcal{A}_{O,H}$
```

same time. Finding accessible and coaccessible states can be done in at most $2 \times |\mathcal{A}_{O,H}|$. 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 $Ke \times |\mathcal{P}| \times |H| \times 2^{|\mathcal{P}|}$. The size of the resulting summary increases the size of $\mathcal{A}_{O,H}$ by at most $Ke \times |\mathcal{P}| \times |H| \times 2^{|\mathcal{P}|}$ states. □

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

Corollary 3 *Computing online a summary for an observation O can be done in*

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

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

Let us compare the respective complexities of the naive online diagnosis mentioned at the beginning of this section (that is $O(|H| \times \sum_{i=1..|O|} i^{|\mathcal{P}|\times|\mathcal{P}_{Obs}|}))$ and this incremental diagnosis. At first sight, incremental construction of a diagnosis seems inefficient, as it has an overhead in $O(|H| \times \sum_{i=1..|O|} d^{h(i-1)} \times |H| \times |\mathcal{P}| \times 2^{|\mathcal{P}|})$. 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 paths need to be extended. Hence, the actual number of paths in the summary at step i should remain lower than $d^{h(i)}$. Note also that $h(i)$ is necessarily lower than $|H| \times i$. 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 $O\left(|O| \times (d^K + d^K \times |H| \times |\mathcal{P}| \times 2^{|\mathcal{P}|})\right)$.

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 the diagnoser guarantees that each transition of the diagnosis HMSC becomes safe after a finite number of observations. Last, notice that the diagnosis HMSC obtained from a summary built online is exactly the diagnosis obtained in the offline setting, and hence with the same number of nodes in $O\left(|H| \times |O|^{|\mathcal{P}|\times|\mathcal{P}_{Obs}|}\right)$.

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 $a.(a'.b)^i$, then the unsafe part kept

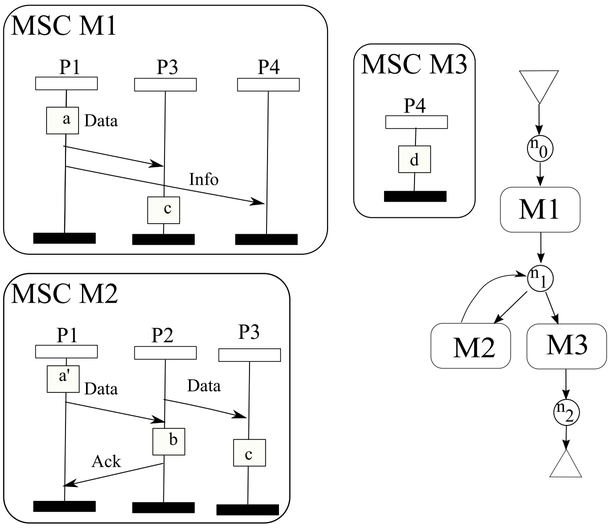

Fig. 15 An example HMSC in which online observation can grow unboundedly

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. Hence, if $P1$ and $P2$ are faster than $P3$, or if $P3$ needs more time than other processes to report occurrences of c, the unsafe part of the diagnosis 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.

6.4 Online diagnosis with finite memory

We have shown in previous sections that offline and online diagnosis is always a decidable problem for HMSCs. However, resources limitations may impose to perform online diagnosis with finite memory. Formally, for a given model H, we want an online algorithm that always runs with finite memory, and outputs $\mathcal{A}_{O,H}$ for any observation O. 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 [33]. 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 an 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 an 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.* $V = \{(\phi(e), \phi(e')) \mid (e, e') \in \mu_M\}$

An 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 $O(2^{|\mathcal{P}|\times|N|} \times (b \times |N| \times |\mathcal{P}|)^{|\mathcal{P}|})$. 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 “acknowledgment” of 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 an 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 sets of configurations from $\mathcal{B}_H$.

However, observing a regular system 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 send 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 of r events per time unit, then online diagnosis with finite memory is achievable.

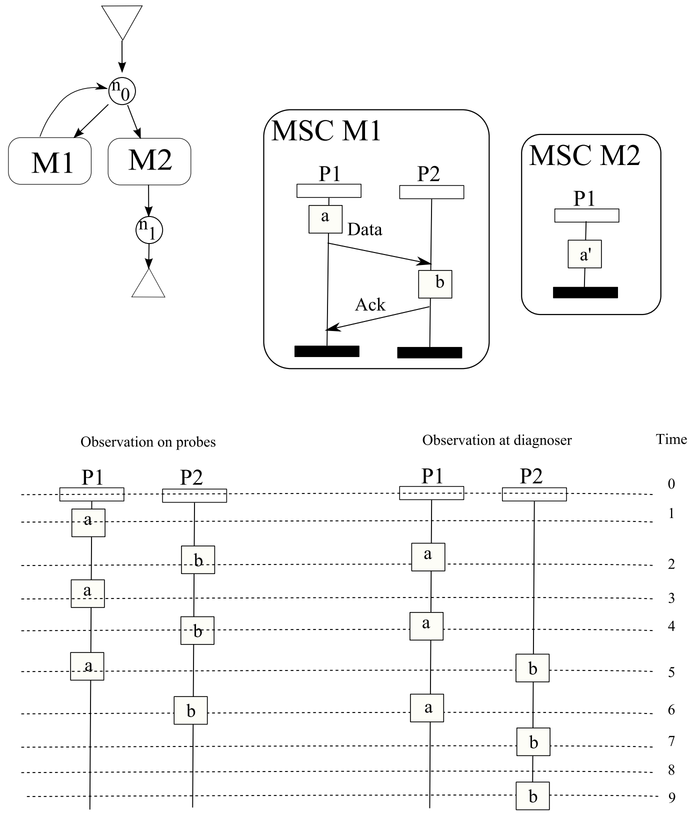

Fig. 16 Desynchronization of observation at probes and at diagnoser

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 $O(NO \times k \times 2^{|\mathcal{P}|\times|N|} \times (b \times |N| \times |\mathcal{P}|)^{|\mathcal{P}|})$, and in time lower than*

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

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

Proof: The main idea is to monitor the system with the finite automaton $\mathcal{B}'_H$ that describes the projection of all linearizations of H, plus a finite set of partially ordered events that are not yet explained. However, the trick is to forget these memorized 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 last node of H visited, and the part of each visited MSC that have not yet been executed. At each 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 is 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 the latter 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 has not yet been 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 $f \leq e$, but not the converse. Hence, one may have to wait up to δ_{Obs} time units to receive f, and then forget e. In the meantime, the observer may observe the arrival of $\delta_{Obs} \times r$ 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 diagnoser to recall at most $\delta_{Obs} \times r \times |\mathcal{P}|$ events. The diagnoser 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 needs 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 been reported to the diagnoser, the explanations (linearizations) computed so far may explain the whole observation or only up to some $k \leq n$, with $k > n - |\mathcal{P}| \times \delta_{obs} \times r$ and have to memorize unexplained events. Hence, at a given moment, one may have to remember up to $NO \times 2^{|\mathcal{P}| \times |N|} \times (b \times |N| \times |\mathcal{P}|)^{|\mathcal{P}|}$ states, where NO is the number of observations of size at most $|\mathcal{P}| \times \delta_{obs} \times r$. Kleitman and Rotschild [21] gave a bound of $2^{\frac{n^2}{4} \times \frac{3n}{2} \times \ln n}$ for the number of partial orders of size n. We must in addition label these orders, i.e. NO is in $O\left(\sum_{i \in 1..k} 2^{\frac{i^2}{4} \times \frac{3i}{2} \times \ln i} \times |\Sigma_{obs}|^i\right)$, for $k = |\mathcal{P}| \times \delta_{obs} \times r$ □

From Theorem 4, it is straightforward to design a diagnosis algorithm. We keep in memory states of the form (s, M, i), where s is a state of $\mathcal{B}'_H$ and M a partially ordered set of events, and $i \in \mathbb{N}$ is the step at which this state was recorded. Intuitively, i means that s is remembered since i^{th} observed event. We start the diagnosis from the initial state of $\mathcal{B}'_H$, and an empty order, and at date 0. When a new event e arrives at a date $d(e)$, for each state (s, M, i) 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 $(s, M, i) \xrightarrow{\alpha(e)} (s', M, i+1)$. 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 it will appear later in the linearization that currently stops at s. So, e is kept in memory with its date of observation $d(e)$, that is we transform state (s, M, i) into $(s, M.e, i)$
- every state with memorized event with date lower than $d(e) - \delta_{obs}$ is declared wrong and forgotten.

At the end of the observation, one can rebuild all transitions, and discard the paths 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 6.3 with a regular HMSC, we might be unable to forget safe parts.

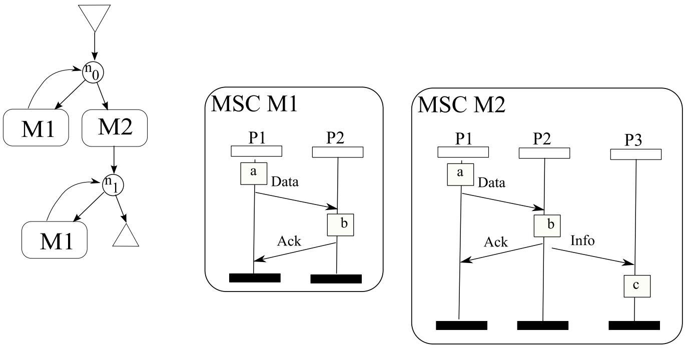

Fig. 17 A regular HMSCs that may generate small modules

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 HMSC of Figure 18, where the HMSC node reached in states q_0, q_1, q_3, q_5 is n_0, and n_1 in q_2, q_4, q_6. 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 improved. This situation holds for observations of any size that do not contain an occurrence of event c. Hence, even if we 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. Now, if we consider an observation $O' = O \circ \{c\}$ that is simply the observation O of Figure 18 with an additional event labeled c, then the online diagnoser needs only to remember that the diagnosis built from O' ends in H at node n_1. The erased part of the diagnosis will contain all paths with three occurrences of a and b, and one occurrence of c ending in n_1. Conversely, a linearization based algorithm memorizes a state from an automaton that recognizes the language $(a.b)^*.c$ plus a finite number of occurrences of a and b, so it runs with finite memory. An interesting extension of this work is to study conditions when one algorithm should be preferred to the other.

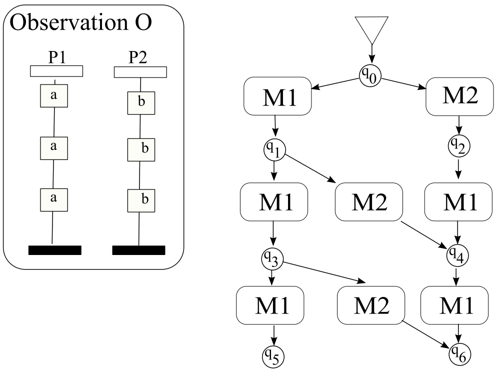

Fig. 18 The diagnosis for an observation O.

6.5 Online detection of existence

Solving the existence problem offline consists in building a diagnosis HMSC as described in section 5 , and verifying that the language of this HMSC is not empty. We now want to check existence online. Formally, given H and successive observations $O_0 = M_\epsilon$ and $O_{i+1} = O_i.e_i, i \in \mathbb{N}^+$, where e_i it the i^{th} event arriving at the diagnoser, we have to check at every step $i \in \mathbb{N}^+$ that $\mathcal{A}_{O_i,H}$ contains at least one explanation, and if not raise an alarm. 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 next page.

6.6 Summary of results

Let us briefly summarize the results obtained so far:

Offline diagnosis: we have shown (Theorem 1, section 5) that offline diagnosis can be performed in space in $O(|H| \times |O|^{|\mathcal{P}| \times |\mathcal{P}_{Obs}|})$, and in Corollary 2 that this can be done with time complexity in $O((m + p^2 + (m + p^2)^4) \times d \times |H| \times |O|^{|\mathcal{P}| \times |\mathcal{P}_{Obs}|})$. This offline diagnosis can be split into $\frac{|P_{obs}| \times (|P_{obs}|-1)}{2}$ smaller problems (Theorem 3, section 5.2), each of them returning an answer $\mathcal{A}_{p,q}$ of size in $O(|O|^{2|\mathcal{P}|} \times |H|)$. However, these diagnosis are local to a pair of processes, and one needs to compute a product $\mathcal{A}_{\bigotimes}$ of local diagnosis to obtain a global result. This product is exactly the diagnosis obtained in a global setting, so there is no complexity gain to expect in the worst case.

Offline existence: As offline existence consists in checking emptiness of $\mathbb{L}_{H,\mathcal{A}_{O,H}}$, the worst case complexity is the same as for offline diagnosis. Note

Algorithm 2 OnlineExistence()

```
INPUT : an HMSC $H$, continuously arriving events (denoted $e$)
/* it is assumed that a user can interrupt the procedure at any time */
/* and then consider $\mathcal{A}_{OH}$ as the explanation for all observed events */
OUTPUT : an error message if no diagnosis exists for the observation collected */
or a diagnosis $\mathcal{A}_{OH}$ if the algorithm is interrupted
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
        /* there is no explanation containing $e$, and no explanation can be */
        /* found with more observed events: existence checking can stop*/
        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."
```

however that the construction of $\mathbb{L}_{H,\mathcal{A}_{O,H}}$ can be stopped as soon as a witness path reaching a final state, and hence showing the non-emptiness of $\mathbb{L}_{H,\mathcal{A}_{O,H}}$ is found. In most cases, if the construction of $\mathcal{A}_{O,H}$ is performed with a depth first policy, the existence problem is likely to be solved faster than the diagnosis. Existence can also be split into several sub-problems. However, existence of a solution for every pair of observable processes does not guarantee that a global solution involving all processes exists. However, $\mathbb{L}_{H,\mathcal{A}_{O,H}}$ is empty as soon as some $\mathbb{L}_{H,\mathcal{A}_{p,q}}$ is empty. Similarly building incrementally the product of local diagnosis $\mathcal{A}_{p_1,q_1} \otimes \dots \mathcal{A}_{p_n,q_n}$, one can stop as soon as the language $\mathbb{L}_{H,\mathcal{A}_{p_1,q_1} \otimes \dots \mathcal{A}_{p_i,q_i}}$ is empty for some $i < n$, without computing the whole product. Hence, splitting the existence problem is likely to improve efficiency in practice. A major improvement can be expected if diagnosis can be made local, that is if we can compute a local diagnosis $\mathcal{A}_p$ for each observed process and ensure that $\bigotimes\limits_{p \in \mathcal{P}_{Obs}} \mathcal{A}_p = \mathcal{A}_{O,H}$ for every observation O. Of course, if all observations are located on one single process p, computing $\mathcal{A}_p$ is sufficient. However, beyond trivial cases such as this one, we have not yet found syntactic conditions on H and Σ_{Obs} ensuring for every observation O that $\mathcal{A}_p = \mathcal{A}_{O,H}$, or that the diagnosis for O is given by a product $\bigotimes\limits_{p \neq q \in X} \mathcal{A}_{p,q}$ for a small subset $X \subset \mathcal{P}_{Obs}$. Combining smaller diagnosis problems is also the key to distributed diagnosis frameworks, and is an interesting research direction.

Online diagnosis: for the online diagnosis, the memory needed is of course in $O(|H| \times |O|^{|\mathcal{P}| \times |\mathcal{P}_{Obs}|})$, as the diagnosis computed online and offline must be the same, and as one can never be sure that even a single safe part will appear during the construction of the diagnosis. The increment time for online diagnosis (the time needed to compute $\overline{\mathcal{A}_{O.e,H}}$ from $\overline{\mathcal{A}_{O,H}}$) is in $O(|\overline{\mathcal{A}_{O,H}}| + Ke \times |\mathcal{P}| \times |H| \times 2^{|\mathcal{P}|})$, where Ke is the number of transitions in $\mathcal{A}_{O,H}$ that lead to an extension of $\mathcal{A}_{O,H}$ (Proposition 11, section 6.3). This means that the overall worst case time complexity to build a diagnosis online is in $O\left(\sum\limits_{i\in 1..|O|} |H| \times (i-1)^{|\mathcal{P}|\times|\mathcal{P}_{Obs|}} + d^{h(i-1)} \times |H| \times |\mathcal{P}| \times 2^{|\mathcal{P}|}\right)$, where $h(i)$ is the height of the summary built at step i by the algorithm. Hence, worst case complexity is not favorable to online diagnosis. However, in practice, an observable event is unlikely to remain unobserved forever, and safe parts should appear. If the height of the memorized diagnosis remains bounded by some constant K, then for a HMSC of degree d the memory needed is in $O(d^K)$, and the overall complexity of online diagnosis becomes in $O\left(|O| \times (d^K + d^K \times |H| \times |\mathcal{P}| \times 2^{|\mathcal{P}|})\right)$.

Online existence: Online existence has the same space and time complexity as online diagnosis, as one may have to remember the whole diagnosis HMSC built from the beginning if no safe part appears.

Diagnosis with finite memory: for the regular HMSCs, when a delay δ_{obs} between an observation and its reception by the diagnoser exists, and when processes execute events with a bounded rate r one can use an automaton recognizing all linearizations of H to monitor a system. In such case, online diagnosis can be performed with memory at most in $O(NO \times k \times 2^{|\mathcal{P}|\times|N|} \times (b \times |N| \times |\mathcal{P}|)^{|\mathcal{P}|})$, where $k = |\mathcal{P}| \times \delta_{obs} \times r$ and NO is in $O\left(\sum\limits_{i\in 1..k} 2^{\frac{i^2}{4}\times\frac{3i}{2}\times \ln i} \times |\Sigma_{obs}|^i\right)$. At each step, the diagnoser knows in which states the system can be and computes the next reachable states. The past states and transitions need not be kept in memory, and can be stored. The time and space complexities for diagnosis with regular HMSCs are in $O(|O| \times NO \times k \times 2^{|\mathcal{P}|\times|N|} \times (b \times |N| \times |\mathcal{P}|)^{|\mathcal{P}|})$. Note that in the regular online case, we memorize linearizations of the HMSC model. Hence, regularity should not be exploited in an offline context, where the non-interleaved compact representation of runs provided by HMSCs is fully exploited.

Online existence with finite memory: In the regular setting, one can simply memorize the states in which the system can be. This means that the algorithm runs with memory and space in $O(NO \times k \times 2^{|\mathcal{P}|\times|N|} \times (b \times |N| \times |\mathcal{P}|)^{|\mathcal{P}|})$, as there is no need to record past states. The worst case time complexity however is in $O(|O| \times NO \times k \times 2^{|\mathcal{P}|\times|N|} \times (b \times |N| \times |\mathcal{P}|)^{|\mathcal{P}|})$.

Optimality: Note that there is some minimal amount of information that needs to be stored by a diagnoser, independently from the technique used and from the nature (regular or not) of the HMSC model. The diagnoser needs to memorize at least all final transitions appended to the $\mathcal{A}_{O,H}$ built so far, and for each transition, the set E_{keep} of observed events that may still have a causal successor (there are at least $|\mathcal{P}_{obs}|$ such events) plus an abstraction of the causal relations between events, which can be represented as a function from E_{keep} to $2^{\mathcal{P}}$. Keeping a trace of this causal ordering is essential to make sure that observed causal dependencies appear in the provided explanations. So when one possesses a bound on the maximal width w of $\mathcal{A}_{O,H}$ and on the number of events kp that need to be memorized, then a diagnoser can not run faithfully with memory lower than in $O(w \times kp \times |\mathcal{P}|)$. Note however that such bounds w and kp are not guaranteed to exist in general.

Let us now discuss the optimality of the offline and online diagnosis constructions. We already know from Remark 2 that the diagnosis generated for an observation O and a model H are not minimal, as they generate **all** paths embedding O, including paths which contain a prefix that already embeds O. One can also remark that the information recorded in states of the generators that are constructed by the offline algorithm is not minimal: states contain information on the respective ordering of events (more precisely using function g) that is not always useful. In some cases, the fact that an event $e \in E_O$ is "seen" by a process p (i.e. $e \in g(p)$) could be forgotten, hence saving space at construction time. However, this simplification can be done only if it is sure that the considered event e will not have an observable causal successor on process p. In the offline context, where all observed successors of an event e are known at diagnosis time, the size of the diagnosis HMSC can also be reduced after full construction of the diagnosis. We can ignore the information contained in states, and use a co-determinization procedure to ensure that the diagnosis is the minimal structure generating all paths that embed the observation. However, we can co-determinize only when final states are known, so this technique does not apply in an online setting. Similarly, pertinence of the information recorded is states highly depends on event that may occur in the future after the current observation. As these events are not known in advance, all available information attached to nodes of the diagnosis must be kept, even if it turns out to be useless in the future. Even worse, useless information may have to be kept forever, so the diagnosis built online is not even asymptotically minimal, as one may never be able to decide which part of the recorded information will become useless.

The offline diagnosis algorithm has been implemented in a scenario tool called SOFAT [16], and was tested on a TCP-IP scenario model, using randomly generated observations. The TCP-IP model can be found in [13]. Though the theoretical worst case bound for offline and online diagnosis is rather pessimistic, the algorithm returns solutions rapidly for observations with hundreds of observable events. Note that carefully choosing the events that need to be observed (for instance making sure that there exists one observable event per MSC) helps reducing the number of paths considered in $\mathcal{A}_{O,H}$ hence saving space and time. However, diagnosis in real distributed systems such as telecommunication networks needs to address gigabytes of logged information on each monitored site. To scale up to this size, diagnosis will probably need to lose some precision. The solutions might be to forget a part of the built diagnosis, and return only a partial answer, to filter logs, hence loosing some precision, or also to work with coarse grain logs that is group sequences of events and work with a smaller quotiented observation.

7 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*.

Many intrusion detection systems (IDS) are "signature-based", that is recognize 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 attack databases have been collected [28, 11].

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 still relies on monitoring, but compares an observation with a description of a normal situations. 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 [31], or with a specification-based approach. [22] proposes a logical framework to define normal behaviors, and [34] 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 [34] 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 [2,19,20] for more information.

In this section, we show that the diagnosis techniques described in 5 and 6 can be used for anomaly detection. The advantages when using scenarios are manifold. First, scenarios are partial order models, and can avoid costly interleaved representations. Second, scenarios 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.

7.1 Monitoring architecture

The anomaly detection framework proposed hereafter relies on the diagnosis techniques proposed in sections 5 and 6, and on an architecture that collects a subset of what occurs in the system, and compares the observation to the HMSCs defining normal behaviors. We can formalize the problem as follows: given an observation O and a set of HMSCs $H_1, \dots, H_k$ over disjoint alphabets $\Sigma_1, \dots, \Sigma_k$, depicting legal uses of a system, check for every $i \in 1..k$ that there exists an explanation for $\Pi_{\Sigma_i}(O)$ provided by H_i.

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 an HMSC H_{ij}, and define an observation alphabet Σ_{ij}. We set $H_{ij} = (N_{ij}, \longrightarrow_{ij}, n^i_{ij}, , \mathcal{M}_{ij}, F_{ij})$, where all MSCs in $\mathcal{M}_{ij}$, are defined over a subset of $\mathcal{P}_{ij} = U_i \cup \mathcal{P}$. More intuitively, the depicted interactions 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 session 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 $\bigcup\limits_{i \in 1..q, j \in 1..k} \Sigma_{ij}$ 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.

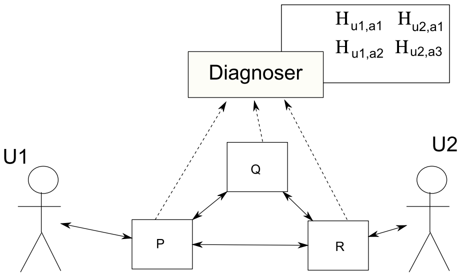

Fig. 19 Architecture of the anomaly detection framework

7.2 Anomaly detection with Diagnosis techniques

The anomaly detection framework proposed in this section uses several elements of 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 $\leq_{O'} = \big(\{(e,e') \in E^2_{O'} \cap \leq_O \mid \phi(e) = \phi(e')\} \cup (E^2_{O'} \cap <_O)\big)^*$*

Intuitively, the restriction is the transitive and reflexive closure of the total ordering due to processes plus the ordering due to messages among events which label belongs to Σ_{ij}. Let us now show how the existence problem described in section 5.1 can be used for anomaly detection. Clearly, if we consider that an HMSC model provided to the diagnoser represents *all* legal behaviors, proving that no explanation exists for an observation O means that O can not be considered as a legal behavior of the system. Furthermore, in a distributed setting, checking existence can be fast, as no explanation exists for an observation as soon as one sub-problem has no solution.

We propose a partial order anomaly detection framework based on the architecture and techniques of sections 5 and 6. 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.

Let us now formalize our scenario-based anomaly detection framework. First of all, we can notice that if we use an 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 that attackers 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 $O(k \times q \times h \times |O|^{(n+1)\times p_{obs}})$, 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 over a set of processes of size lower or equal to n, plus a user. Then, a diagnosis has to be performed for every H^*_{ij}, that is at most $q \times k$ 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. Even when an execution of the monitored system is not a run of the model, the observation recorded during this execution may still find a compatible explanation. As a consequence, the detection framework may miss some faulty behaviors.

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, and raise an alarm 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 a 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.

8 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, which opens the way for distributed diagnosis frameworks. An easier diagnosis related problem called the existence is 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 diagnoser, 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 [36] for instance.

Another interesting extension of this work is to consider diagnosis from more powerful scenario models such as Causal MSCs [13]. 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 a scenario based anomaly detection framework that detects an anomaly when the existence problem has a negative answer. 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. Our intuition is that known attacks can also be described as finite abstract scenarios over roles instead of processes. 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 some applications such as diagnosis can be entirely handled 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. P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artif. Intell.*, 110(1):135–183, 1999.
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. J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theor. Comput. Sci.*, 37:77–121, 1985.
6. Daniel Brand and Pitro Zafiropoulo. On communicating finite state machines. Technical Report 1053, IBM Zurich Research Lab., 1981.
7. L. Castellano, G. De Michelis, and L. Pomello. Concurrency versus interleaving: an instructuve example. *Bulletin of the EATCS*, 31:12–14, 1987.

8. Th. Chatain and C. Jard. Symbolic diagnosis of partially observable concurrent systems. In *FORTE'04*, volume 3235 of *Lecture Notes in Computer Science*, pages 326–342. Springer, 2004.
9. Th. Chatain and C. Jard. Time supervision of concurrent systems using symbolic unfoldings of time petri nets. In *FORMATS'05*, volume 3829 of *Lecture Notes in Computer Science*, pages 196–210, 2005.
10. O. Contant, S. Lafortune and D. Teneketzis. Diagnosability of discrete event systems with modular structure. *Discrete Event Dynamic Systems*, 16(1):9–37, 2006.
11. DARPA. Intrusion detection dataset. http://www.ll.mit.edu/IST/ideval/data/data_index.html, 2000.
12. C. Fidge. Logical time in distributed computing systems. *Computer*, 24(8):28–33, 1991.
13. Th. Gazagnaire, B. Genest, L. Hélouët, P. S. Thiagarajan and S. Yang. Causal message sequence charts. *Theor. Comput. Sci.*, 410(41):4094–4110, 2009.
14. S. Genc and S. Lafortune. Distributed diagnosis of discrete-event systems using petri nets. In *ICATPN*, volume 2679 of *Lecture Notes in Computer Science*, pages 316–336, 2003.
15. 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.
16. L. Hélouët. Sofat : Scenario formal analysis toolbox. INRIA Rennes, 2008. www.irisa.fr/distribcom/Prototypes/SOFAT/.
17. L. Hélouët, T. Gazagnaire, and B. Genest. Diagnosis from scenarios. In *Workshop on Discrete Event Systems, WODES'06*, 2006.
18. ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)*. ITU-TS, September 1999.
19. A. K. Jones and R. S. Sielken. Computer system intrusion detection: A survey. Technical report, Dept. of Computer Science, University of Virginia, 1999.
20. P. Kabiri and A. A. Ghorbani. Research on intrusion detection and response: A survey. *International Journal of Network Security*, 1(2):84–102, 2005.
21. D. J. Kleitman and B. L. Rotschild. Asymptotic enumeration of partial orders. *Transactions of the American Mathematical Society*, (205):205–220, 1975.
22. 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.
23. F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems*, 4:197–212, 1994.
24. F. Mattern. On the relativistic structure of logical time in distributed systems. *Parallel and Distributed Algorithms*, pages 215–226, 1989.
25. A. Muscholl. Matching specifications for Message Sequence Charts. In *FoSSaCS'99*, LNCS 1578, pages 273–287, 1999.
26. A. Muscholl and D. Peled. Message sequence graphs and decision problems on mazurkiewicz traces. In *MFCS*, pages 81–91, 1999.
27. A. Muscholl, D. Peled, and Z. Su. Deciding properties for message sequence charts. In *FOSSACS'98*, pages 226–242. Springer-Verlag, 1998.
28. University of California. Kdd cup 1999 data, 1999.
29. OMG. Uml superstructure specification, v2.0. OMG Document number formal/05-07-04, 2005.
30. Y. Pencolé and M. O. Cordier. A decentralized model-based diagnostic tool for complex systems. *International Journal on Artificial Intelligence Tools*, 11(3):327–346, 2002.
31. O. Salem, S. Vaton, and A. Gravey. An efficient online anomalies detection mechanism for high-speed networks. In *MonAM'07*, 2007.
32. 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.
33. F. B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.
34. 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.

35. R. Su, W. M. Wonham, J. Kurien, and X. Koutsoukos. Distributed diagnosis for qualitative systems. In *in 6th International Workshop on Discrete Event Systems, Zaragoza (WODES-2002*, pages 169–174, 2002.
36. S. Yovine. Model checking timed automata. In *European Educational Forum: School on Embedded Systems*, pages 114–152, 1996.