A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks

Yannick Pencolé [a] Marie-Odile Cordier [b]

[a] *CSL, the Australian National University, ACT 0200, Australia*

[b] *IRISA / Université de Rennes 1, Campus de Beaulieu, 35000 Rennes, France*

Abstract

We address the problem of diagnosing large discrete event systems. Given a flow of observations from the system, the goal is to explain these observations on-line by identifying and localising possible failures and their consequences across the system. Model-based diagnosis approaches deal with this problem but, apart very recent proposals, either they require the computation of a global model of the system which is not possible with large discrete event systems, or they cannot perform on-line diagnosis. The contribution of this paper is the description and the implementation of a formal framework for the on-line decentralised diagnosis of such systems, framework which is based on the "divide and conquer" principle and does not require the global model computation. This paper finally describes the use of this framework in the monitoring of a real telecommunication network.

Key words: Model-based diagnosis, Discrete event systems, Decentralised model, Distributed artificial intelligence, Telecommunication networks, Fault propagation

1 Introduction

The problem we deal with is the supervision of complex and large discrete event systems such as telecommunication networks, electricity distribution network, and more generally speaking *Immobots* [29]. Given a supervision

Email addresses: Yannick.Pencole@anu.edu.au (Yannick Pencolé), cordier@irisa.fr (Marie-Odile Cordier).

system continuously receiving observations (alarms) sent by the system components, our purpose is to help operators to identify failures. Two classical approaches in monitoring such systems are knowledge-based techniques that directly associate a diagnosis to a set of symptoms, for example expert systems [17], or chronicle recognition systems [9,7], and model-based techniques which rely on a behavioural model of the system [22]. The main weakness of the first approach is the lack of genericity: as the system changes (new components, new connections, new technologies), a new expertise has to be acquired. Therefore, we focus on model-based techniques which are known to be better suited to that kind of system than expertise-based approaches.

A number of model-based approaches for diagnosing discrete event systems have been proposed in both the AI and control engineering literature. They cover continuous-variable systems which, after quantisation, are represented as discrete systems [15], as well as "discrete by nature" systems such as communicating processes which exchange messages and alarms. The majority of these approaches are centralised approaches [15,23,26]. For instance, the diagnoser approach [26] consists in the compilation of diagnostic information in a data structure (called *diagnoser*), which maps observations to failures for on-line diagnosis. The main drawback of centralised approaches is that they require to explicitly build the global model of the system which is unrealistic for large, complex systems such as telecommunication networks.

The considered systems are naturally distributed so it is easier to model those systems in a decentralised way. An approach for diagnosing discrete event systems using decentralised diagnosers can be found in [8], but the computation of each decentralised diagnoser is still based on a global model. There also exist methods relying on a decentralised model [2,6], but these are used *off-line* to solve a diagnosis problem *a posteriori*. Recently, due to the need of solving a diagnosis problem on-line, a monitoring-based approach [13,14] has been developed: this method mixes a diagnoser approach [26] with an extended version of the decentralised model of [2] by computing on-line only the interesting parts of a centralised diagnoser without computing a global model. This method still has the problem that it systematically uses global states of the system which can be a problem when dealing with large discrete event systems.

In this paper, we propose a formal framework providing an approach which relies on a decentralised model and computes on-line diagnosis of large discrete event systems. Firstly, we propose a formalism for decentralised models based on communicating automata. This formalism allows us to model behaviours of large discrete event systems in a modular way and to use decentralised algorithms on it thanks to a generic synchronisation operation.

Secondly, we define the diagnosis problem inside this framework and propose

an algorithm to make on-line diagnosis. To make an on-line diagnosis system, efficiency is the key issue. The idea is to split the flow of observations into temporal windows. For each temporal window, we compute a diagnosis for a subsystem (*subsystem diagnosis*) and then we build a diagnosis of the whole system (*global diagnosis*) by *merging* these subsystem diagnoses. The merging operation is applied thanks to an original strategy which dynamically recognises an efficient way to apply the merging operation based on the observations of the current temporal window.

The paper is organised as follows. We first introduce the type of systems that we consider, the monitoring problem, and a small example which will be used as an illustration throughout the paper (sections 2 and 3). In section 4, we present the formalism based on communicating automata, which is used to represent in a decentralised way the model of the system, and a synchronisation operation which allows us to perform *decentralised reasonings* on any subpart of the system in the same way. Section 5 explains the diagnostic task by defining observations and diagnoses and in section 6, we formally present the decentralised diagnosis approach and prove its equivalence with respect to the centralised one in the proposed framework. Section 7 focuses on the choices about the implementation of the decentralised diagnosis approach in order to apply the approach on-line. Firstly, partial order reduction techniques are shown to be well-suited for efficiently representing the diagnoses. Secondly, the merging operation strategy, taking into account the interactions of the subsystem diagnoses dynamically, is proved to greatly improve the efficiency of the global diagnosis computation. In section 8, the incremental aspect of the diagnosis problem is introduced and is shown to be essential in the context of dynamical system monitoring. Section 9 presents some results relying on a real case of telecommunication network. This study has been done in the context of the MAGDA project [1] and demonstrates the benefits of a decentralised approach. Finally, section 10 presents related work and section 11 concludes this paper and discusses several perspectives relying on the presented work.

2 Monitoring large reactive discrete event systems

2.1 System characteristics

A typical system is depicted in Figure 1: *components* communicate each other with the help of *communication channels*. A *component* is an entity that has

[1] RNRT project MAGDA: funded by the French Ministère de la Recherche; the other partners of this project are France Telecom R&D, Alcatel, Ilog, and Paris-Nord University.

a finite set of *internal states*. The system is event-driven, *i.e.* it evolves with the occurrence of *events* on the components.

An *exogenous event* (event from the set Σ_{exo}) is an event produced by the environment of the system. Due to the fact that events are instantaneous (they do not have delay), the probability that two events produced by the environment occur at the same time is practically null, hence the following hypothesis.

Hypothesis 1 *Two exogenous events cannot occur at the same time on the system.*

Such an event may trigger a change of state in one component. During that state change, the affected component may produce *communication events* (event from the set Σ_{com}) towards its neighbourhood by emitting *messages* via its communication channels and also produce *observable events* (event from the set Σ_{obs}) towards the environment of the system by emitting *observable messages*. The reception of a message from a communication channel is also a communication event and may change the internal state of the component which receives it. In that case, the component affected by this event may also emit communication and observable events.

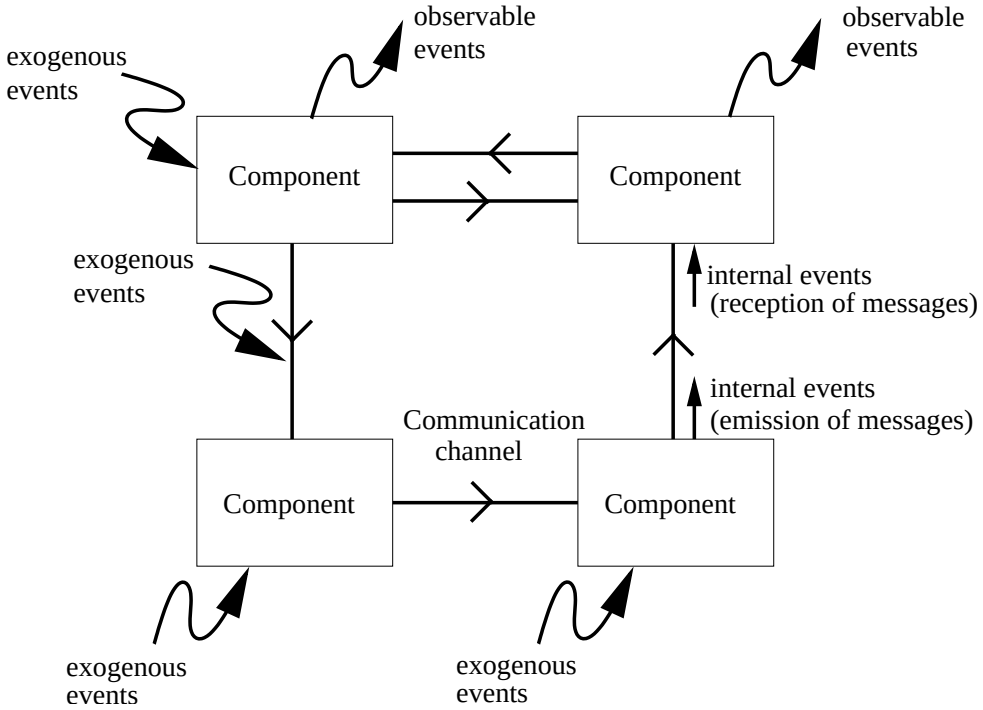

Fig. 1. Reactive discrete event system

A communication channel respects the following assumption which guarantees that the system has a finite set of states.

Hypothesis 2 *A communication channel between two components is bounded.*

A communication channel can be of any type: among the channel types, queues are especially considered, like for example:

- *instantaneous queue*: such a queue has no buffer, the emission of a message from a component and the reception of the same message to the destination is the same event;
- *first in first out* queue (FIFO): such a queue has a bounded buffer, the messages conveyed by this queue are received in the same order they have been emitted;
- queue with *loss of messages*: such a queue is not reliable, conveyed messages can be lost due to several types of problems (saturation of the buffer, loss due to the occurrence of an exogenous event on the channel which affects its behaviour...).

2.2 Monitoring of the system

In order to help the human agent (or *supervisor*) in charge of managing the system, *i.e.* detecting failures and deciding reconfiguration/repair actions, a *supervision system* is needed: its task is to record *observations* of the supervised system and to analyse them in order to produce a concise view of the state and the history of the system for the supervisor.

2.2.1 Observability of the system

Definition 1 (Observation) *An* observation *is the reception, at a given date, by the supervisor, of a message sent by a component of the supervised system.*

Any observation corresponds to the emission of an observable message by a component. The message of an observation is supposed to contain an information about the component which has emitted it, it follows that the supervisor knows about the component source of every observation.

Because the system is large, the supervisor may not be located next to the supervised components. In the majority of cases, an *observation channel* has to be considered between any component which emits *observable messages* and the supervisor. As a consequence, the emission of an observable message is not necessarily the same event as the reception of this message by the supervisor.

The observation system which is the set of the observation channels respects the following hypothesis.

Hypothesis 3 *The observation system is complete, reliable and efficient.*

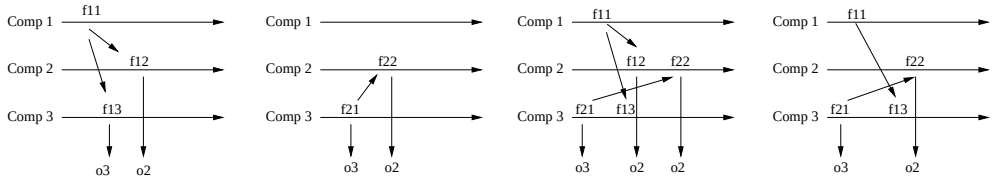

Fig. 2. Failure propagations and interferences

The completeness of the observation system means that for every kind of observable message there exists an *observation channel* which can convey this kind of message. The reliability of the observation system means that the observation channels do not loose messages. Every message emitted by the components are effectively received by the supervisor. The efficiency means that any message in an observation channel is conveyed efficiently without message overtaking. Consequently, in the following, we can assume that any observation channel is an instantaneous or a reliable FIFO bounded queue.

2.2.2 Monitoring task

The purpose of the monitoring task is to detect, localise, and identify problems that occur on the system. These problems can be physical (an equipment is down, a cable is cut) or logical (a station is rebooting, a logical connection is down...). Our purpose is not to explain in details what is happening on the system but only what a supervisor agent needs to know. In the following, we will consider that a *failure* is any occurrence of an event which is considered as pertinent for the supervisor in the sense that he wants to trace the occurrences of this event. For most of the failures on the considered systems, there are some automatic recovery procedures (*recovery events*), so failures can disappear (*intermittent failures*). A failure that do not have any recovery event is called a *permanent failure*.

One of the main difficulties in the monitoring of large discrete event systems is that the occurrence of a *primary failure* on a component may have effects (called *secondary failures*) on other components. As a consequence, the occurrence of a failure on one component (or one communication channel) may cause the occurrence of several secondary failures in the whole system and the reception of a huge number of observations by the supervisor. There could be also several failure propagations at the same time which can interfere and provide a huge number of possible observation sets depending on the way the propagations interfere with each other. Moreover, it is also possible that a secondary failure occurs depending on the way the different propagations interfere. Because of those interferences, we need not only to identify the first causes of a problem (the primary failures) but also the way they interfere to also identify the secondary failures. The recovery events have also to be diagnosed because they are also part of interferences in the failure propagations.

Example 1 *In Figure 2, four different failure propagations are presented. The two figures on the left describe two single failure propagations inside a system composed of three components. f11 and f21 are the primary failures and they occur respectively on component 1 and component 3. The consequences of f11 are the occurrences of the secondary failures f12 and f13 and the emissions of the observations o3 o2. The consequences of f21 are the occurrence of the secondary failure f22 and the emission of the observations o3 o2. In the figures on the right, f11 and f21 occur both and their respective propagations interfere: it can happen that the secondary failures are not the same. For example, if f22 occurs then the failure f12 may not occur even if f11 has occurred. This is due to the nature of f22 and f12 (for instance if f22 is "power down of the machine" and f12 is "reboot of the machine", when the power is down the machine cannot reboot but the power can go down when the machine is rebooting).*

Another monitoring problem is the fact that the observations that are generally emitted when a failure occurs can be *masked* because of the occurrence of another failure in the past. The consequence of the *masking phenomenon* is the fact that it increases the number of failures that can occur without observable consequences and, therefore, the number of possible explanations for a given set of observations.

Example 2 *In Figure 2, on the right side, when f13 occurs on the component 3, no observation is emitted like in the other figures. The failure f21 has masked the observation o3 that should have been emitted after f13. In that example, if we observe o3 o2, there are three possible explanations. The single propagations (on the left side) and the multiple failure propagation of the right side.*

3 Example

This section describes a small example of supervised system that is used as an illustration of the different ideas presented in this paper (see Figure 3). In the following, this example is referred as *Toynet*. This system is composed of three data switches (*SW1*, *SW2*, *SW3*). These switches are in charge of emitting and receiving data in a ring network. Two switches *SWi* and *SWj* communicate each other with the help of the connection *cnij*. Each switch *SWi* is managed by a control station *CSi*.

Here is the behaviour description of the supervised system. A switch transmits data through two connections: a west connection (for *SW1*, it is *cn12*) and an east connection (for *SW1*, it is *cn31*). A connection between two switches is considered as a bidirectional communication channel. This communication

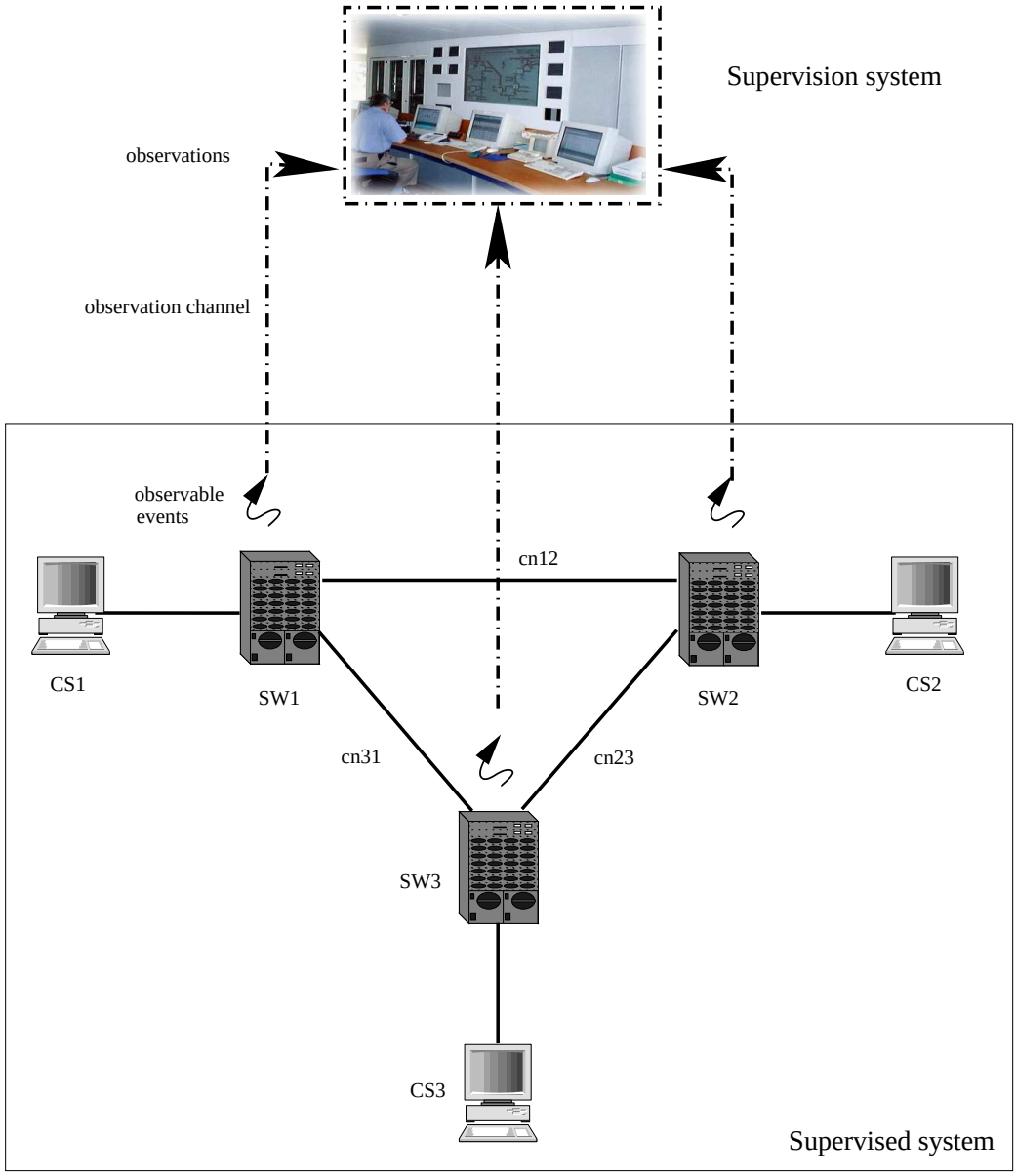

Fig. 3. Telecommunication network example and its supervision system.

channel is not reliable and can be affected by the *cut* failure. If the connection is cut (*cutCnij*), *SWi* emits an observable event (for example, if *cn12* is cut (*cutCn12*), *SW1* emits the observable event *SW1cn12* and if *cn31* is cut (*cutCn31*), *SW1* emits the observable event *SW1cn31*). Then, the switch goes to its waiting mode. If the connection is reestablished (*workCnij*), the switch goes back to its normal mode. A switch can break down (*SW1brk*), in this case, an observable event mechanism informs the supervision system by the emission of the observable event *SWidown*. Moreover, the control station *CSi* detects this problem and tries to reinitialise the switch *SWi* (*SWireboot*). After a reinitialisation (*SWiendreboot*), the switch is operational again and emits an observable event *SWiok*. Two kinds of failures can happen on a control sta-

tion. Firstly, the station can hang up (*CSioff*) and then recovers a normal mode (*CSion*). When the station recovers a normal mode, an observable event *CSiok* is emitted. This observable event is conveyed via the switch *SWi*, so the observable event is masked if the switch is not in its normal mode. A station can also reboot (*CSireboot)* and at the end of the reinitialisation (*CSiendreboot*), an observable event *CSiok* is emitted. The communication channel between a control station and its switch is considered as reliable and instantaneous.

As far as the supervision system is concerned, each switch is connected to it via an observation channel. In this example, for the sake of simplicity, those observation channels are instantaneous queues, *i.e* the emission of an observable message by a switch (an observable event) corresponds exactly to the reception of this message by the supervision system (the observation).

4 Decentralised model of the system

As said in the introduction, we decided to use model-based approaches which are recognised to be better suited to systems that can evolve (new components, new technologies). Due to the great number of components, it is quite unrealistic to rely on a global model of such systems. This section explains how the model of the system is described in a decentralised way by means of local models, which describe the behaviours of each component and each communication channel of the system and a generic synchronisation operation which describes the way the local models interact each other.

The formalism used to model a system is based on the formalism defined in [24]. In this former article, the authors propose to model a component as a communicating automaton which represents the way *messages* are received or emitted via a set of *ports* belonging to the component. The model of the system is then represented as a set of communicating automata and a set of *links*. A link is an association between an output port (port from where messages are emitted) and an input port (port where messages are received) which defines synchronisation rules between components (the emission and the reception of a message on a link is synchronised). [2] From a practical point of view, this formalism is very intuitive and allows to model the system in a modular and hierarchical way. For the sake of simplicity and without loss of generality, [3] we present in this paper an abstraction of this formalism. In this formalism, the

[2] A link does not correspond to a communication channel previously described. In the formalism of [24], a communication channel is represented like a component. See section 4.2 for details.

[3] The temporal aspects (delays) that are defined in [24], are nevertheless not considered at all in this paper.

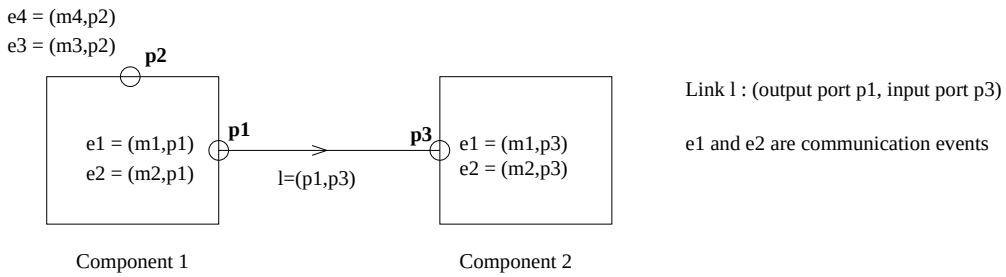

Fig. 4. Abstraction of ports, messages and links

notions of *port*, *message* and *link* are abstracted with the help of the notion of *event*: an event is the reception or the emission of a message via a port. If a port is linked to another port, then the emission of a message from the output port and the reception of the same message by the associated input port is represented in our formalism by a *communication event* (see Figure 4).

Although we present the abstracted version of our formalism for the sake of simplicity, in practice we use the non-abstracted version so that we can benefit of the modularity and the hierarchical way of modelling a system.

4.1 Model of a component

A *component* c_i receives two kinds of events:

(1) *exogenous events* Σ^i_{exo}, events from the environment ($\Sigma^i_{exo} \subseteq \Sigma_{exo}$);
(2) *communication events* $\Sigma^i_{com\_rcv}$, reception of messages coming from other components of the system ($\Sigma^i_{com\_rcv} \subseteq \Sigma_{com}$).

Hypothesis 4 *A component c_i cannot receive two different events from $\Sigma^i_{exo} \cup \Sigma^i_{com\_rcv}$ at the same time.*

A component can also emit two kinds of events:

(1) *observable events* Σ^i_{obs}, emission of messages that can be observed by a supervision system ($\Sigma^i_{obs} \subseteq \Sigma_{obs}$);
(2) *communication events* $\Sigma^i_{com\_emit}$, emission of messages to other components of the system ($\Sigma^i_{com\_emit} \subseteq \Sigma_{com}$).

Note 1 *Because of the way the abstraction is defined, a communication event is involved only in two components: the sender of the message and the receiver of the message (see Figure 4).*

Definition 2 (Model of a component) *The model of the* component c_i *is*

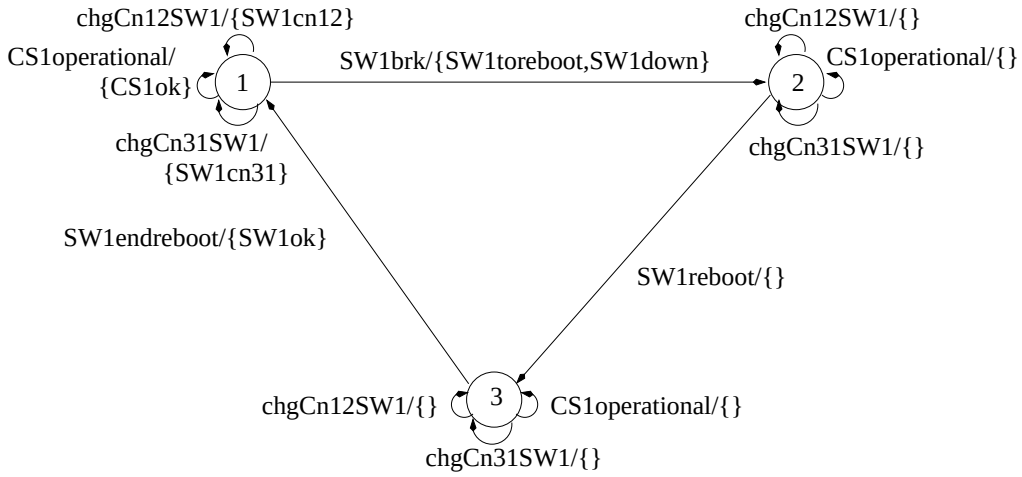

Fig. 5. Model of the control-part of *SW1* (noted *SW1ctl*).

described by the communicating finite state machine:

$$\Gamma_i = (\Sigma^i_{rcv}, \Sigma^i_{emit}, Q_i, E_i)$$

- Σ^i_{rcv} *is the set of received events* ($\Sigma^i_{rcv} = \Sigma^i_{exo} \cup \Sigma^i_{com\_rcv}$);
- Σ^i_{emit} *is the set of emitted events* ($\Sigma^i_{emit} = \Sigma^i_{obs} \cup \Sigma^i_{com\_emit}$);
- $\Sigma^i_{rcv} \cap \Sigma^i_{emit} = \emptyset$;
- Q_i *is the set of component states;*
- $E_i \subseteq (Q_i \times \Sigma^i_{rcv} \times 2^{(\Sigma^i_{emit})} \times Q_i)$ *is the set of transitions.*

Note 2 *For any component transition* $q \xrightarrow{t} q'$*, we will note by* $rcv(t)$ *the event from* Σ^i_{rcv} *which triggers the transition* t*,* $emit(t)$ *the set of events emitted by* t*, and among the events of* $emit(t)$*,* $obs(t)$ *the set of observable events.*

The model of the control-part of the component *SW1* (noted *SW1ctl*) is depicted on Figure 5 (transitions are noted $q \xrightarrow{rcv(t)/emit(t)} q'$). The failure exogenous events are: *SW1brk* (*SW1* begins to break down), *SW1reboot* (*SW1* begins to reboot) and *SW1endreboot* (*SW1* terminates its reboot). The received communication events are: *CS1operational* (reception of a message *"the control station becomes operational"*) and *chgCn12SW1*, *chgCn31SW1* (reception of a message *"the status of a connection has changed"*). Among the emitted events, here are the observable ones: *SW1down*, *SW1ok*, *SW1cn31*, *SW1cn12* and *CS1ok*. There is also one emitted communication event: *SW1toreboot* (emission of a message *"the switch has to reboot"*).

4.2 Model of a communication channel

As said in section 2.1, the components are connected by any kinds of bounded communication channels. In the case where the channel is not an instanta-

neous queue between two components (i.e. in the case where the emission and the reception of a message are not instantaneous) or in the case where the channel is not reliable, we need to model the behaviour of the channel. In that case, our proposal is to model the communication channel like a component by using a communicating automaton. Every emission of message from a component c_1 to a component c_2 corresponds to the reception of this message by a communication channel between c_1 and c_2 and the reception of the message by the component c_2 corresponds to the emission of the same message by that channel. If a failure occurs on a channel, that failure changes the internal state of the channel and may disturb the transmission of messages between components.

Example 3 *In* Toynet*, the connection cnij is considered as a component. This component can receive two failure events: cutCnij (the connection is cut) and workCnij (the connection is reestablished). The communication channel between a control station and a switch is not considered as a component because this channel is instantaneous and reliable.*

In the following, without loss of generality, no distinction will be made between component and communication channel: their model are both based on a communicating automaton. The notation c_i will refer to the i^{th} component (or channel) of the system and the notation Γ_i will refer to the model of c_i.

4.3 Model of a system

In this subsection, the decentralised model of a system $\Gamma = \{c_1, \ldots, c_n\}$, where each behaviour of component c_i is represented by a model Γ_i, is formally given. Before defining the model of a supervised system, we formally define the model of one of its subsystems.

4.3.1 Model of a subsystem

A *subsystem* is a set of k components $\gamma = \{c_{i_1}, \ldots, c_{i_k}\}$ from the system where $k \leq n$ and $i_j \in \{1, \ldots, n\}$.

Definition 3 (Model of a subsystem) *The model of the* subsystem $\gamma = \{c_{i_1}, \ldots, c_{i_k}\}$ *is the set of automata* $\{\Gamma_{i_1}, \ldots, \Gamma_{i_k}\}$.

Based on the previous definition of a subsystem, several sets of events are introduced. Σ^{γ}_{rcv} is the set of *received events* of the subsystem γ:

$$\Sigma^{\gamma}_{rcv} \triangleq (\bigcup_{j\in\{1,\ldots,k\}} \Sigma^{i_j}_{rcv}) \setminus (\bigcup_{j\in\{1,\ldots,k\}} \Sigma^{i_j}_{emit}).$$

There are two types of received events: Σ^{γ}_{exo} is the set of exogenous events that occur in γ ($\Sigma^{\gamma}_{exo} \triangleq \Sigma^{\gamma}_{rcv} \cap \Sigma_{exo}$) and $\Sigma^{\gamma}_{com\_rcv}$ is the set of communication events received by γ whose source is a component which is not in γ ($\Sigma^{\gamma}_{com\_rcv} \triangleq \Sigma^{\gamma}_{rcv} \setminus \Sigma^{\gamma}_{exo}$).

Σ^{γ}_{emit} is the set of *emitted events* of the subsystem γ:

$$\Sigma^{\gamma}_{emit} \triangleq (\bigcup_{j\in\{1,\ldots,k\}} \Sigma^{i_j}_{emit}) \setminus (\bigcup_{j\in\{1,\ldots,k\}} \Sigma^{i_j}_{rcv}).$$

In the set of emitted events Σ^{γ}_{emit} is included the set Σ^{γ}_{obs} of *observable events* emitted by γ. The emitted events that are not observable correspond to communication events towards components that do not belong to the subsystem and are noted $\Sigma^{\gamma}_{com\_emit}$.

Σ^{γ}_{int} is the set of *internal events* of the subsystem γ:

$$\Sigma^{\gamma}_{int} \triangleq (\bigcup_{j\in\{1,\ldots,k\}} \Sigma^{i_j}_{emit}) \cap (\bigcup_{j\in\{1,\ldots,k\}} \Sigma^{i_j}_{rcv}).$$

An internal event in the subsystem γ is a communication event associated to two components of the subsystem: it belongs to the set of emitted events of the first component and to the set of the received events of the second one. The set $\{\Sigma^{\gamma}_{rcv}, \Sigma^{\gamma}_{emit}, \Sigma^{\gamma}_{int}\}$ is a partition of the events occurring in γ.

4.3.2 Synchronisation operation in a subsystem

The model of a subsystem represents the propagation of failure events (exogenous events) inside the subsystem as well as events emitted by components that do not belong to the subsystem. It is possible to compute the explicit behaviour of the subsystem thanks to a synchronisation operation applied to the component models $\{\Gamma_{i_1}, \ldots, \Gamma_{i_k}\}$ of the subsystem.

The synchronisation operation is based on a transition system product [1]. As it is done in [1] and for the sake of simplicity in the product definition, some *null transitions* (noted $q \xrightarrow{\mathbf{e}|\{\}} q$) are systematically added to each state q of each communicating automaton. Such a transition means that a component may stay on a given state while other components evolve (asynchronism). Given those transitions, the behaviour of a subsystem can be exhaustively represented by a synchronised product, even if the subsystem has finite asynchronous behaviours.

Definition 4 (Free product) *The* free product *of m communicating automata $T_i = (I_i, O_i, Q_i, E_i), i \in \{1, \ldots, m\}$ is the communicating automaton (I, O, Q, E) such that:*

- $I = I_1 \times \ldots \times I_m$;
- $O = O_1 \times \ldots \times O_m$;
- $Q = Q_1 \times \ldots \times Q_m$ *is the set of states;*
- $E = E_1 \times \ldots \times E_m$ *is the set of transitions*

$$(q_1, \ldots, q_m) \xrightarrow{(t_1,\ldots,t_m)} (q'_1, \ldots, q'_m) = (q_1 \xrightarrow{t_1} q'_1, \ldots, q_m \xrightarrow{t_m} q'_m).$$

In the following, such a product will be noted by $\langle T_1, \ldots, T_m \rangle$. By definition of this product, $\langle T_1, \ldots, T_m \rangle$ is isomorphic to $\langle T_{j_1}, \ldots, T_{j_m} \rangle$ where $\{j_1, \ldots, j_m\}$ is a permutation of $\{1, \ldots, m\}$.

Definition 5 (Synchronised transition) *Given $\{\Gamma_{i_1}, \ldots, \Gamma_{i_k}\}$ the model of the subsystem γ, the transition $q \xrightarrow{t} q' = (q_{i_1} \xrightarrow{t_{i_1}} q'_{i_1}, \ldots, q_{i_k} \xrightarrow{t_{i_k}} q'_{i_k})$ of the product $\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k} \rangle$ is* synchronised *iff:*

- t_j *is null for all* $j \in \{i_1, \ldots, i_k\}$, *or*
- *the three following conditions hold:*

(1) $\exists t_j, j \in \{i_1, \ldots, i_k\}$ *such that* $rcv(t_j) \in \Sigma^\gamma_{rcv}$;
(2) $card(\{t_j, j \in \{i_1, \ldots, i_k\} | rcv(t_j) \in \Sigma^\gamma_{exo}\}) \leq 1$;
(3) for each j *of* $\{i_1, \ldots, i_k\}$ *such that* t_j *is not null,*
  (a) $\forall e \in emit(t_j) \cap \Sigma^\gamma_{int}, \exists l \in \{i_1, \ldots, i_k\} | e = rcv(t_l)$;
  (b) $\forall rcv(t_j) \in \Sigma^\gamma_{int}, \exists l \in \{i_1, \ldots, i_k\} | rcv(t_j) \in emit(t_l)$.

Condition 1 means a synchronised transition $q \xrightarrow{t} q'$ can only be triggered by a set of received events on the subsystem γ. Condition 2 means that, among these events, only one can be exogenous (event from Σ^γ_{exo}) in accordance with the hypothesis 1. The conditions 3.a and 3.b describe the synchronisation rules for internal events inside γ occurring in $q \xrightarrow{t} q'$. If, in $q \xrightarrow{t} q'$, an internal event is emitted by an automaton Γ_j of γ towards another automaton Γ_l of γ, this event has to appear as a received event of Γ_l in $q \xrightarrow{t} q'$ and vice-versa. These conditions represent a propagation of events in the subsystem γ.

A synchronised transition $q \xrightarrow{t} q'$ is thus associated to the sets of events:

(1) $Rcv(t)$ is the set of received events which triggers the transition;
(2) $Int(t)$ is the set of internal events that occur in γ when the transition is triggered;
(3) $Emit(t)$ is the set of emitted events that are emitted outside γ when the transition is triggered, among them, $Obs(t)$ is the set of observable events.

Note 3 *In the following, a synchronised transition* $q \xrightarrow{t} q'$ *will be sometimes written as follows:*

$$q \xrightarrow{Rcv(t)/Int(t)Emit(t)} q'.$$

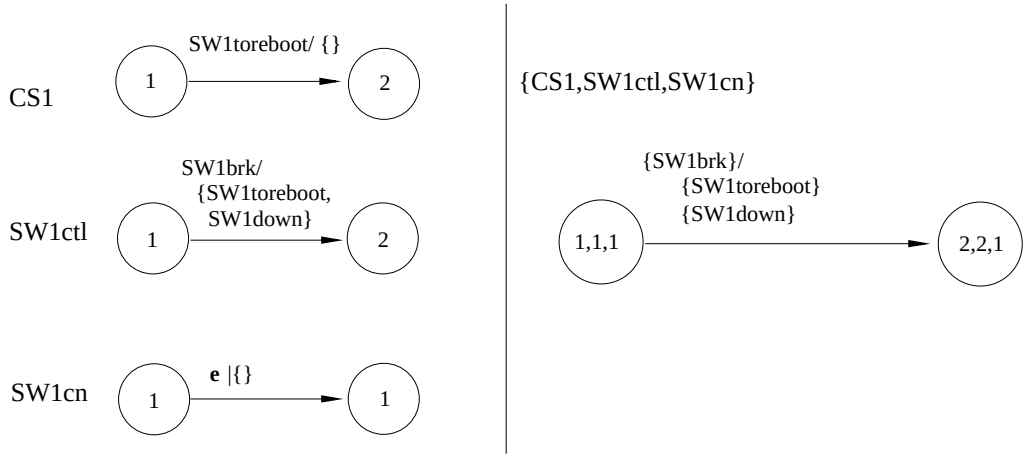

Fig. 6. Synchronised transition in the subsystem {*CS1, SW1ctl, SW1cn*}.

Example 4 *Figure 6 shows an example of synchronised transitions in the subsystem* {*CS1, SW1ctl, SW1cn*} *of Toynet. The event SW1brk is exogenous (the switch breaks down). When this event occurs, an event SW1toreboot is produced between SW1ctl and CS1 (the control station CS1 knows that the switch SW1 has to reboot). An observable event SW1down is also emitted. The connection-part of the switch SW1 (called SW1cn) triggers a null transition. The transition on the right is the synchronisation of the three transitions in the left.*

Based on this notion of synchronisation, the *behaviour* of the subsystem γ can be formally defined as follows.

Definition 6 (Behaviour of the subsystem) *The* explicit behaviour *of the subsystem* $\gamma = \{c_{i_1}, \ldots, c_{i_k}\}$ *is the finite state machine* (I, O, Q', E') *from the free product* $\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k} \rangle = (I, O, Q, E)$ *such that* $Q' \subseteq Q$ *is the set of states and* $E' \subseteq E$ *is the set of synchronised transitions of* E.

In the following, the result of the synchronised product of the automata $\{\Gamma_{i_1}, \ldots, \Gamma_{i_k}\}$ will be noted by $\|\Gamma_{i_1}, \ldots, \Gamma_{i_k}\|$. By extension, we will also denote the explicit behaviour of every subsystem γ by $\|\gamma\|$ where $\{\Gamma_{i_1}, \ldots, \Gamma_{i_k}\}$ is the model of γ.

By definition, the behaviour of a subsystem only composed of one component c_i is the communicating automaton Γ_i itself: every transition t of any automaton Γ_i respects the conditions of a synchronised transition ($Rcv(t) = \{rcv(t)\}$, $Int(t) = \emptyset$, $Emit(t) = emit(t)$ and $Obs(t) = obs(t)$), hence $\Gamma_i = \|\Gamma_i\| = \|c_i\|$.

The behaviour $\|\gamma\|$ of any subsystem γ is a communicating automaton and

can be thus considered like the model of a component. Moreover, a subsystem, being defined by a set of components, can also be defined by a set of subsystems $\gamma_1, \ldots, \gamma_m$. Modelling a subsystem by a set of component models or by a set of subsystem behaviours is equivalent because of the following property on the automata synchronisation.

Theorem 1 *Let γ_1 and γ_2 be two disjoint subsystems, then*

$$\|\gamma_1 \cup \gamma_2\| = \|\|\gamma_1\|, \|\gamma_2\|\|.$$

PROOF. See appendix A.

This property shows that the synchronisation is an associative and commutative operation. Considering any partition of the set of components that defines a subsystem γ, each partition element is also a subsystem. The behaviour $\|\gamma\|$ can be obtained by synchronising the behaviours from every subsystem that the partition defines.

In the following, we will also use the notion of path in a subsystem defined as follows.

Definition 7 (Transition path) *A* transition path *P in a subsystem γ is a sequence (possibly infinite) of consecutive transitions of $\|\gamma\|$.*

In the following, $|P|$ will denote the length of P if P is finite and the value ∞ otherwise.

4.3.3 Decentralised model of the system

The system $\Gamma = \{c_1, \ldots, c_n\}$ is a particular subsystem. This subsystem receives only events from the environment and only emits observable events. Here is the definition of its model.

Definition 8 (Model of the system) *The model of the system Γ is the model of the subsystem $\{c_1, \ldots, c_n\}$ modelled by $\{\Gamma_1, \ldots, \Gamma_n\}$.*

The behaviour of Γ, noted $\|\Gamma\|$, is called the *global model* of the system. By definition, every synchronised transition of the global model is triggered by one exogenous event (see conditions 1 and 2 in the synchronised transition definition) and expresses the consequences of this event inside the system (emission of observable events, change of internal state).

Example 5 Toynet *is modelled by a set of 12 components: one per control-station, one per connection between switches and two per switches (the control-*

part of the switch SWictl and the connection-part of the switch SWicn). The global model of Toynet *is a communicating automaton which contains 8000 states and 76000 transitions.*

5 Diagnosis of the system

5.1 Observable behaviour

From the model, we can define the *observable behaviour* of any subsystem γ. Informally, the *observable behaviour* corresponds to the set of all the sequences of observable events that the subsystem can emit towards the supervision system. Here is the formal definition.

Definition 9 (Observable behaviour) *Given* $P = q_1 \xrightarrow{t_1} \dots \xrightarrow{t_m} q_{m+1} \dots$ *a path of transitions from* $\|\gamma\|$*, the observable behaviour of* P *(noted* $Obs_\gamma(P)$*) is the partially ordered set of observable events produced by* P*. The corresponding partial order relation is defined as follows:*

$$\forall i \in \{1, \dots, |P|\}, \forall j \in \{1, \dots, i-1\}, \forall o_j \in Obs(t_j), \forall o_i \in Obs(t_i), o_j \prec o_i.$$

The observable behaviour *of the subsystem* γ *is the set of observable behaviours from all the paths of* $\|\gamma\|$*.*

By definition, the observable behaviour of the system is the observable behaviour of the subsystem Γ.

Example 6 *Figure 7 presents the observable behaviour corresponding to a path* P*. In this example, the emission of* $o2$ *and* $o3$ *is not ordered. The observable behaviour corresponds to two possible observable sequences:* $o1o2o3o4$ *or* $o1o3o2o4$*.*

5.2 Observed behaviour

An observation is the reception by the supervision system of a message emitted by the system through an observation channel. Thus, every observation corresponds to an observable event of the system. The problem is that, because of the existence of observation channels between the supervised system

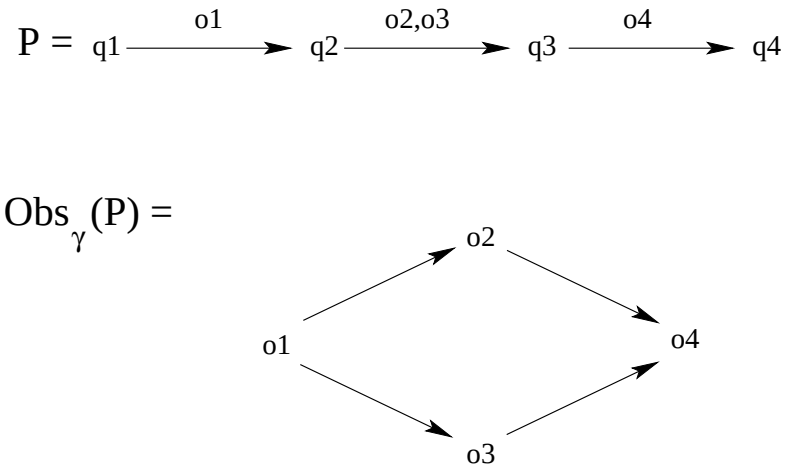

Fig. 7. Observable behaviour of a path P.

and the supervision system, the order of reception of messages by the supervision system is not necessarily the order of the emissions of these messages by the system. In other words, given the sequence σ_γ of received observations from any subsystem γ, a set of emission orders are possible, depending on the nature and the number of the observation channels between the subsystem γ and the supervision system.

In order to deal with this problem, two solutions are possible. In the first one, the set of observation channels is modelled with the supervised system with the help of communicating automata just as we do for communication channels. In that case, the observable behaviour of the model corresponds exactly to the observed behaviour, the difference between emission and reception from observation channels being described inside the model. The problem of this solution is that the size of the model can dramatically increase. The second solution consists in guessing the order of emission on-line. In that case, it is not necessary to model the observation channels. Given their properties (message propagation delays inside an observation channel, potential synchronisations between two observation channels,...), it is possible to define a partial order relation between two observations received from γ so that we know the possible orders of their emission by γ. In the following, such a solution is considered.

Definition 10 (Observed behaviour) *Given the sequence σ_γ of received observations from any subsystem γ, the* observed behaviour $O_\gamma = (\sigma_\gamma, \prec_\gamma)$ *of the subsystem γ is a partially ordered set composed of the observations of σ_γ with a partial order relation $\prec_\gamma$ on them.*

The partial order relation is induced by the characteristics of the observation channels on a subsystem. With the help of the hypothesis 3, if we consider any subsystem γ only composed of one component emitting observable events, then γ is associated with only one instantaneous or bounded FIFO observation channel. In that case, O_γ is totally ordered. Moreover, the partial order relation

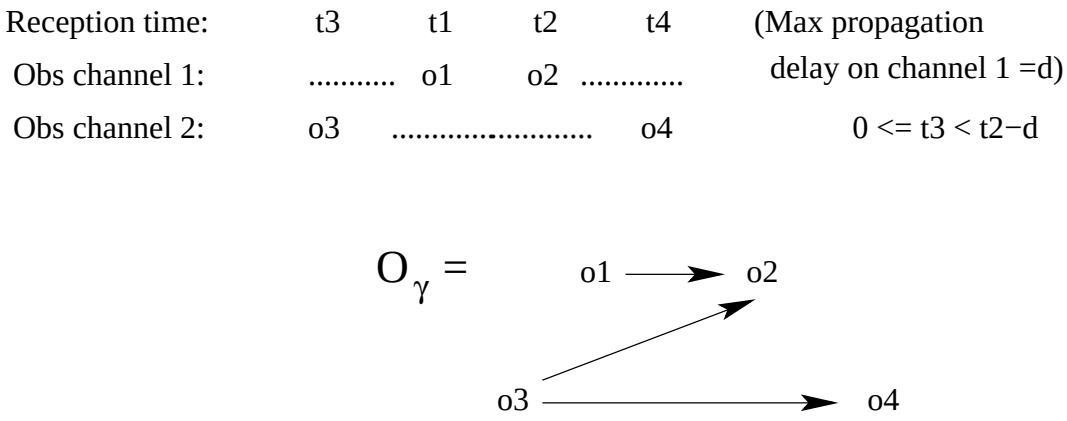

Fig. 8. Observed behaviour O_γ.

on observations has the following properties: given two disjoint subsystems γ_1 and γ_2 and their respective relations $\prec_{\gamma_1}$ and $\prec_{\gamma_2}$, given the relation $\prec_{\gamma_1 \cup \gamma_2}$ on the subsystem $\gamma_1 \cup \gamma_2$, we have:

$$\forall o, o' \in O_{\gamma_1}, o \prec_{\gamma_1} o' \Rightarrow o \prec_{\gamma_1 \cup \gamma_2} o'$$

$$\forall o, o' \in O_{\gamma_2}, o \prec_{\gamma_2} o' \Rightarrow o \prec_{\gamma_1 \cup \gamma_2} o'.$$

Some new orders can also be defined between observations from γ_1 and γ_2 by $\prec_{\gamma_1 \cup \gamma_2}$, expressing characteristics between observation channels from γ_1 and γ_2.

Example 7 *Figure 8 depicts an observed behaviour. In this example, let consider that a subsystem γ is observed with the help of two observation channels. The observation channels convey the observations $o1$ and $o2$ (for channel 1), and $o3$ and $o4$ (for channel 2). The received sequence is $\sigma_\gamma = o3o1o2o4$. The two observation channels are FIFO so $o1 \prec_\gamma o2$ and $o3 \prec_\gamma o4$. Moreover, if we know that the maximal propagation delay of channel 1 is d and the times $t2$ (reception of $o2$) and $t3$ (reception of $o3$) are such that $0 \leq t3 < t2 - d$, it follows that $o3 \prec_\gamma o2$.*

5.3 Definition of the diagnosis

As said in section 2.2.2, the diagnosis problem consists in identifying *failure events* (modelled as exogenous events) and their propagations (modelled as sets of communication events) which explain the observed behaviour of the system. Such a failure propagation in the system is represented by a path of transitions from $\|\Gamma\|$. A path explains an observed behaviour if its observable behaviour is compatible with the observed behaviour. This compatibility is defined below as an operator $\diamond$ between two partially ordered sets.

Definition 11 *Given $S_1 = (E, \prec_1)$ and $S_2 = (E, \prec_2)$ two partially ordered*

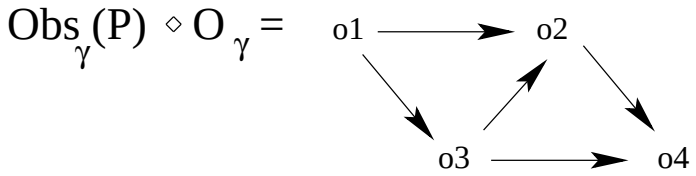

Fig. 9. Joint set $Obs_\gamma(P) \diamond O_\gamma$.

sets, the joint set $S_1 \diamond S_2$ *is the partially ordered set* $(E, \prec_{12})$ *where* $\prec_{12}$ *is recursively defined by*

$$\forall e_1, e_2 \in E, e_1 \prec_{12} e_2 \triangleq$$

$$(e_1 \prec_1 e_2 \vee e_1 \prec_2 e_2 \vee (\exists e_3 \in E | e_1 \neq e_3 \neq e_2 \wedge (e_1 \prec_{12} e_3 \wedge e_3 \prec_{12} e_2)).$$

The joint set of two sets S_1 and S_2 contains the same elements as S_1 and S_2 but the order relation is more restrictive. Informally, the joint set is the partially ordered set whose linear extensions [4] exactly correspond to the intersection of the linear extensions of the observable behaviour and the observed behaviour.

Example 8 *Figure 9 depicts the joint set of* $Obs_\gamma(P)$ *from Figure 7 and* O_γ *from Figure 8.* O_γ *brings a new constraint to* $Obs_\gamma(P)$ $(o3 \prec_\gamma o2)$ *so that the joint set* $Obs_\gamma(P) \diamond O_\gamma$ *represents the unique sequence* $o1o3o2o4$.

Such a set may not exist. Based on the relations $\prec_1$ and $\prec_2$, the relation $\prec_{12}$ is not defined for all $\prec_1$ and $\prec_2$ relations: for example, if $e_1 \prec_1 e_2$ and $e_2 \prec_2 e_1$, $\prec_{12}$ is a relation such that $e_1 \prec_{12} e_2 \wedge e_2 \prec_{12} e_1$, and as a consequence, the relation $\prec_{12}$ is not an order relation (not antisymmetric). In the case where we cannot define an order relation $\prec_{12}$, the relations $\prec_1$ and $\prec_2$ are said to be *incompatible* and $S_1 \diamond S_2$ does not exist.

Example 9 *If in Figure 8, the relation* $o4 \prec_\gamma o2$ *is added, then, because* $o2 \prec o4$ *in* $Obs_\gamma(P)$, O_γ *and* $Obs_\gamma(P)$ *(Figure 7) are incompatible.*

With the help of this operator, here is the formal definition of the diagnosis of a system, called *global diagnosis*.

Definition 12 (Global diagnosis) *Given the decentralised model of the system* Γ, *given* O_Γ *the observed behaviour of the system, the* global diagnosis $\Delta(O_\Gamma)$ *is the set of paths* P *of* $\|\Gamma\|$ *explaining* O_Γ, *i.e. such that* $Obs_\Gamma(P) \diamond O_\Gamma$ *exists.*

[4] A *linear extension* (also called *linearisation*) of a partially ordered set is a sequence of the elements of the set such that if $e_1 \prec e_2$ then e_1 is before e_2 in the sequence.

This definition expresses the fact that a diagnosis is a set of behaviours constrained by the observations O_Γ. Each path of the diagnosis is a possible explanation of the observations. This explanation contains the sequence of failure events that have potentially occurred on the system and their propagations in the system.

5.4 Presentation of the result to the supervisor

The diagnosis, like defined in the previous section, contains the complete and necessary information to understand what has happened in the system. In that sense, a diagnosis can be seen as a database which is updated on-line. However, this information can be too complex to be given to the supervisor on-line. When the supervisor is monitoring the system, the information he needs can be only the list of primary failures for instance. This information is just an abstraction of the complete diagnosis and can be easily computed at the same time.

Once the supervisor wants to deeply analyse the reason why a particular failure has occurred (off-line analysis), he will query for the set of behaviours that could explain the failure. For example, in the MAGDA project (see section 9), this query is implemented with the help of a graphical user interface representing the topology of the supervised network. This interface provides a way to browse the behaviours that are related to the particular failure and to project them in the topology presented by the interface, so that the supervisor is able to see the failure propagations directly on a representation of the topology of the supervised system. The implementation of such an interface is only based on simple searches in graphs and does not need the use of complex algorithms.

5.5 Conclusion

In the framework of supervision of large discrete event systems the diagnosis information has to be rich. Not only the identification of the failures is needed but also their propagations in the system are important because they can explain every emitted alarm. This is particularly true in applications like the supervision of telecommunication networks. As a consequence, the diagnosis must summarise those propagation of failures, it is why such a definition is proposed for the diagnosis of a given dynamic system. This definition, as a set of sequences, can be compared to the definitions given in [2,6].

Because the computed diagnosis information is very rich, an ergonomic interface has to be implemented in order to help the supervisor. The purpose of the interface is to extract pertinent information for on-line analysis (list of

failures,...) and to offer the possibility to deeply analyse off-line the behaviours that are related to the diagnosed failures.

Using a centralised approach like [25,23] or any approach which needs the explicit computation of the global model $\|\Gamma\|$ [8,27] is problematic. Because $\|\Gamma\|$ is based on a Cartesian product, its size is in the worst case exponential to the number of components in the system. Computing the global model of a system which contains more than one hundred components is thus impossible with common computer resources. Thus, using a centralised approach for computing the diagnosis of such a system is impossible due to the intractable size of the global model. It is the reason why we propose an approach which relies on component models and does not require the explicit computation of the global model $\|\Gamma\|$.

6 Decentralised diagnosis approach

The proposed decentralised approach is based on the *divide and conquer* principle. Because the computation of a diagnosis based on the global model is impossible, the problem is divided so that smaller diagnoses are computed on smaller models (component models). These diagnoses are then progressively *merged* to obtain *subsystem diagnoses* and finally the global diagnosis of the system.

6.1 Subsystem diagnosis

The *subsystem diagnosis* definition is a generalisation of the global diagnosis definition. The purpose of the subsystem diagnosis of $\gamma = \{c_{i_1}, \ldots, c_{i_k}\}$ is to explain the set of observations emitted by γ using the model of the subsystem γ.

Definition 13 (Subsystem diagnosis) *Given γ a subsystem, given O_γ the observed behaviour of this subsystem, the* subsystem diagnosis $\Delta_\gamma(O_\gamma)$ *is the set of paths P of $\|\gamma\|$ explaining O_γ, i.e. such that $Obs_\gamma(P) \diamond O_\gamma$ exists.*

Every path of the subsystem diagnosis provides an explanation of the observations from this subsystem. This explanation is *local*, in other words, this explanation does not take into account the behaviour of the components that do not belong to γ. Each explanation makes the hypothesis that every message, exchanged with a component that do not belong to γ, is possible. By definition, the diagnosis of the subsystem Γ is the global diagnosis itself.

As said in the previous subsection, the subsystem diagnoses make the hypothesis that every message exchange is possible between two subsystems. The purpose of the merging operation is to check if such exchanges are possible or not, according to the global model of the system. This check consists in synchronising every path of a subsystem diagnosis with every path of the other subsystem diagnoses. The merging operation is based on the theorem 2. Before presenting this theorem, the notion of path synchronisation is introduced.

Given γ_1, γ_2 two disjoint subsystems and P_1, P_2 two transition paths belonging to $\|\gamma_1\|$ and $\|\gamma_2\|$ respectively, the notation $\|P_1, P_2\|$ will denote the set of paths resulting from the synchronisation of P_1 and P_2. Formally, $\|P_1, P_2\|$ could be obtained firstly by synchronising the set of transitions from $\|\gamma_1\|$ that occur in P_1 with the set of transitions from $\|\gamma_2\|$ that occur in P_2 and secondly by extracting from this synchronised finite state machine the set of paths P such that every transition of P_1 and P_2 is triggered in P in the same order as in P_1 and P_2. By construction, every path of $\|P_1, P_2\|$ is a path of $\|\|\gamma_1\|, \|\gamma_2\|\| = \|\gamma_1 \cup \gamma_2\|$. Moreover, the set of paths $\|P_1, P_2\|$ may be empty; in this case, the paths P_1 and P_2 are not *synchronisable*. The notation is extended to a set of paths $P_1, \ldots, P_m$ on a set of disjoint subsystems $\gamma_1, \ldots, \gamma_m$: $\|P_1, \ldots, P_m\|$ will denote the set of paths resulting from the synchronisation of the paths P_i, the set of paths being obtained in the same manner as the set $\|P_1, P_2\|$.

Theorem 2 *Given γ_1 and γ_2 two disjoint subsystems,*

$$P \in \Delta_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2}) \Leftrightarrow$$
$$\exists P_1 \in \Delta_{\gamma_1}(O_{\gamma_1}) \wedge \exists P_2 \in \Delta_{\gamma_2}(O_{\gamma_2}) \wedge P \in \|P_1, P_2\| \wedge Obs_{\gamma_1 \cup \gamma_2}(P) \diamond O_{\gamma_1 \cup \gamma_2} \textit{ exists}.$$

PROOF.

($\Leftarrow$) $P_1 \in \Delta_{\gamma_1}(O_{\gamma_1})$ and $P_2 \in \Delta_{\gamma_2}(O_{\gamma_2})$ so $P_1 \in \|\gamma_1\|$ and $P_2 \in \|\gamma_2\|$. $P \in \|P_1, P_2\|$, so $P \in \|\gamma_1 \cup \gamma_2\|$ by construction. $Obs_{\gamma_1 \cup \gamma_2}(P) \diamond O_{\gamma_1 \cup \gamma_2}$ exists, therefore $P \in \Delta_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2})$.

($\Rightarrow$) $P \in \Delta_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2})$ so $Obs_{\gamma_1 \cup \gamma_2}(P) \diamond O_{\gamma_1 \cup \gamma_2}$ exists. By definition, P is a path from $\|\gamma_1 \cup \gamma_2\|$. This path can be obtained from the synchronised product $\|\|\gamma_1\|, \|\gamma_2\|\|$ (see theorem 1), thus there are some paths P_1 from $\|\gamma_1\|$ and P_2 from $\|\gamma_2\|$ such that $P \in \|P_1, P_2\|$.

Suppose for the sake of contradiction that $Obs_{\gamma_1}(P_1) \diamond O_{\gamma_1}$ does not exist. Therefore, P_1 explains all the observations of O_{γ_1} but in an order which is incompatible with the order of O_{γ_1}. In other words, there exist at least

two different observations o_1 and o_2 such that $o_1 \prec o_2$ in O_{γ_1} and $o_2 \prec o_1$ in $Obs_{\gamma_1}(P_1)$. If $o_1 \prec o_2$ in O_{γ_1}, then $o_1 \prec o_2$ in $O_{\gamma_1 \cup \gamma_2}$ (definition 10). Moreover, if $o_2 \prec o_1$ in $Obs_{\gamma_1}(P_1)$ then $o_2 \prec o_1$ in $Obs_{\gamma_1 \cup \gamma_2}(P)$ (definition 9). Consequently, $Obs_{\gamma_1 \cup \gamma_2}(P) \diamond O_{\gamma_1 \cup \gamma_2}$ does not exist.

The existence of $Obs_{\gamma_2}(P_2) \diamond O_{\gamma_2}$ can be shown in the same manner. Finally, the existence of $Obs_{\gamma_1 \cup \gamma_2}(P) \diamond O_{\gamma_1 \cup \gamma_2}$ implies the existence of $Obs_{\gamma_1}(P_1) \diamond O_{\gamma_1}$ and $Obs_{\gamma_2}(P_2) \diamond O_{\gamma_2}$, so P_1 and P_2 respectively belong to $\Delta_{\gamma_1}(O_{\gamma_1})$ and $\Delta_{\gamma_2}(O_{\gamma_2})$. □

Corollary 1 *Given* $\{\gamma_1, \ldots, \gamma_m\}$ *a set of subsystems which is a partition of the set of components of the system* Γ*:*

$$P \in \Delta_\Gamma(O_\Gamma) \Leftrightarrow$$
$$(\bigwedge_{i=1}^{m} \exists P_i \in \Delta_{\gamma_i}(O_{\gamma_i})) \wedge P \in \|P_1, \ldots, P_m\| \wedge Obs_\Gamma(P) \diamond O_\Gamma \ \textit{exists}.$$

PROOF. Because of theorem 2,

$$P \in \Delta_\Gamma(O_\Gamma) \Leftrightarrow$$
$$\exists P_{1,\ldots,l} \in \Delta_{\bigcup_{i=1}^{l} \gamma_i}(O_{\bigcup_{i=1}^{l} \gamma_i}) \wedge \exists P_{l+1,\ldots,m} \in \Delta_{\bigcup_{i=l+1}^{m} \gamma_i}(O_{\bigcup_{i=l+1}^{m} \gamma_i})$$
$$\wedge P \in \|P_{1,\ldots,l}, P_{l+1,\ldots,m}\| \wedge Obs_\Gamma(P) \diamond O_\Gamma \ \textit{exists}.$$

The result is obtained by applying recursively the same theorem on $P_{1,\ldots,l}$ and $P_{l+1,\ldots,m}$ and by noticing that, by construction, if two paths P_2 and P_3 respectively belong to $\|P_4, P_5\|$ and $\|P_6, P_7\|$ then a path P_1 belongs to $\|P_2, P_3\|$ iff the path P_1 belongs to $\|P_4, P_5, P_6, P_7\|$. □

6.3 Summary

We have defined a formal framework for the decentralised diagnosis approach (see Figure 10). The model of the system Γ is represented in a decentralised way as a set of communicating automata $\{\Gamma_1, \ldots, \Gamma_n\}$ and a synchronisation operation. The idea is then to compute the diagnosis for each component c_i (corresponding to the more basic subsystems) based on its model Γ_i and then to progressively merge the results in order to obtain diagnoses on bigger subsystems and finally to obtain the global diagnosis. Within this framework, due to the fact that the synchronisation is an associative and commutative operation, we have the guarantee that, by merging the subsystem diagnoses in any order, the result is the same and is the global diagnosis (see corollary 1).

7 On-line diagnosis implementation

This section presents the implementation of the decentralised diagnosis approach based on a decentralised model of the system. In order to make an on-line diagnosis approach, the algorithms must be efficient and based on an efficient representation of the diagnoses. Firstly, partial order reduction techniques are shown to be well-suited for efficiently representing the diagnoses. Secondly, a merging operation strategy, taking into account the interactions of the subsystem diagnoses dynamically, is presented.

7.1 Diagnosis representation

7.1.1 Finite representation

In the framework, the diagnosis $\Delta_\gamma(O_\gamma)$ is defined as a set of paths of transitions of $\|\gamma\|$. A path may be infinite because of an infinite sequence of silent transitions (in the behaviour $\|\gamma\|$, such sequences are represented by loops of unobservable transitions). Because of the merging operation, a finite representation of the diagnosis is needed. This representation is based on a finite state machine which also represents infinite silent sequences by loops. Here is the definition of this representation.

Let $\|\gamma\| = (I, O, Q, E)$ be the behaviour of the subsystem γ. Let σ_γ be a finite observation sequence and $O_\gamma = (\sigma_\gamma, \prec_\gamma)$ be the corresponding observed behaviour. Consider every transition path P from $\Delta_\gamma(O_\gamma)$, P is such that $P = q_1 \xrightarrow{t_1} \ldots \xrightarrow{t_m} q_{m+1} \ldots$ where $q_i \in Q, i \in \{1, \ldots, |P|\}$ and $q_i \xrightarrow{t_i} q_{i+1} \in E, i \in \{1, \ldots, m\}$. Consider a transition $q_i \xrightarrow{t_i} q_{i+1}$ of P and $P_i = q_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{i-1}} q_i$ the sub-path of P from q_1 to q_i, the state q_i can thus be represented by the state $q_i^{finite} = (q_i, Obs_\gamma(P_i) \diamond O_\gamma^i)$ where O_γ^i is the observed behaviour $(\sigma_\gamma^i, \prec_\gamma)$ where σ_γ^i is the prefix sequence of σ_γ such that $Obs_\gamma(P_i) \diamond O_\gamma^i$ exists. Such a set $Obs_\gamma(P_i) \diamond O_\gamma^i$ always exists because $Obs_\gamma(P) \diamond O_\gamma$ exists by definition. q_i^{finite} expresses the fact that the state q_i is a possible current state of $\|\gamma\|$ after the explanation of the observed behaviour O_γ^i. The state q_{i+1} is then associated to the state $q_{i+1}^{finite} = (q_{i+1}, Obs_\gamma(P_{i+1}) \diamond O_\gamma^{i+1})$. The transition $q_i \xrightarrow{t_i} q_{i+1}$ is thus represented by $q_i^{finite} \xrightarrow{t_i} q_{i+1}^{finite}$.

Let Q^{finite} be the set of states q^{finite} defined as above for every path from $\Delta_\gamma(O_\gamma)$, by construction Q^{finite} is a finite set ($Q^{finite} \subseteq Q \times Pr(O_\gamma)$, where $Pr(O_\gamma)$ is the set of partially ordered sets containing a subset of the elements of O_γ). Let E^{finite} be the set of transitions $q_i^{finite} \xrightarrow{t_i} q_{i+1}^{finite}$ defined as above for every path from $\Delta_\gamma(O_\gamma)$, E^{finite} is also finite by construction. Q^{finite} and

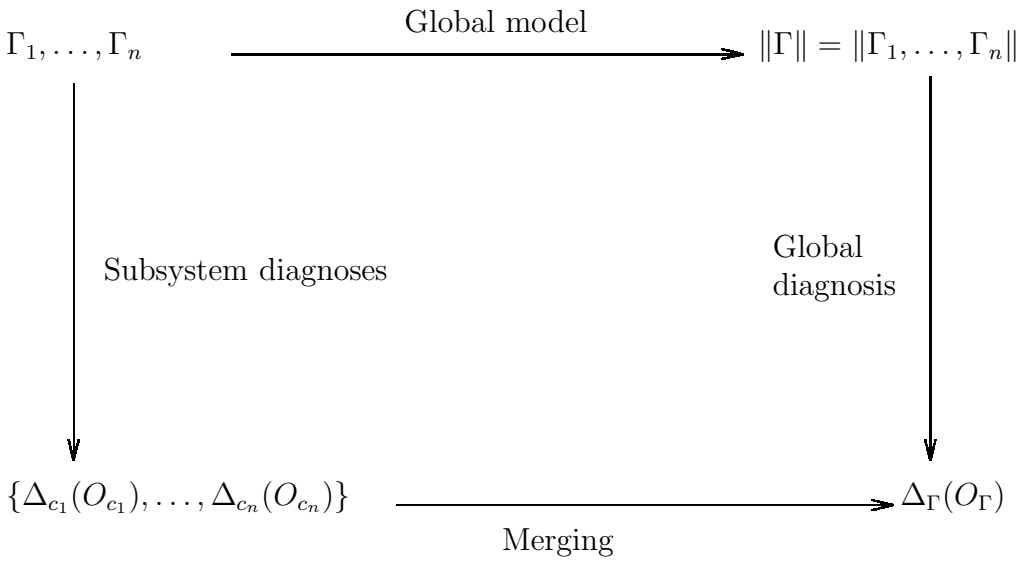

Fig. 10. Centralised / decentralised approach

E^{finite} define a finite representation of the diagnosis $\Delta_\gamma(O_\gamma)$.

Definition 14 (Finite representation) *Let $\|\gamma\| = (I, O, Q, E)$ be the behaviour of the subsystem γ, the* finite representation *of $\Delta_\gamma(O_\gamma)$ is the finite state machine $\Delta_\gamma^{finite}(O_\gamma) = (I, O, Q^{finite}, E^{finite})$ where Q^{finite} and E^{finite} are respectively the set of states and transitions defined above.*

The diagnosis of $\Delta_\gamma(O_\gamma)$ can be represented by $\Delta_\gamma^{finite}(O_\gamma)$ (see Figure 11). The states $q^{finite} \in Q^{finite}$ such that $q^{finite} = (q, \emptyset)$ are called the *initial states* of the diagnosis. According to the observations, the subsystem γ could have been in one of these states q before the emission of any observable event. The states $q^{finite} \in Q^{finite}$ such that $q^{finite} = (q, O)$ with $|O| = |O_\gamma|$ are called the *final states* of the diagnosis. According to the observations, the subsystem γ is in one of these states q.

Nevertheless, the representation has a problem: its size. Each path of $\Delta_\gamma^{finite}(O_\gamma)$ represents a path of diagnosis, i.e. a sequence of events. Because of the distributed nature of the diagnosed systems, a lot of events (failure events) may occur in a concurrent way, so dealing with sequences means enumerating the sequences where a failure event f_1 occurs independently before an event f_2 and where f_2 occurs before f_1. From a diagnosis point of view, because f_1 and f_2 are *independent*, if they occur both, it is not important to know about the order. It is the reason why, a *reduced* representation of the diagnosis has been introduced. This reduction is based on a partial order reduction method [18].

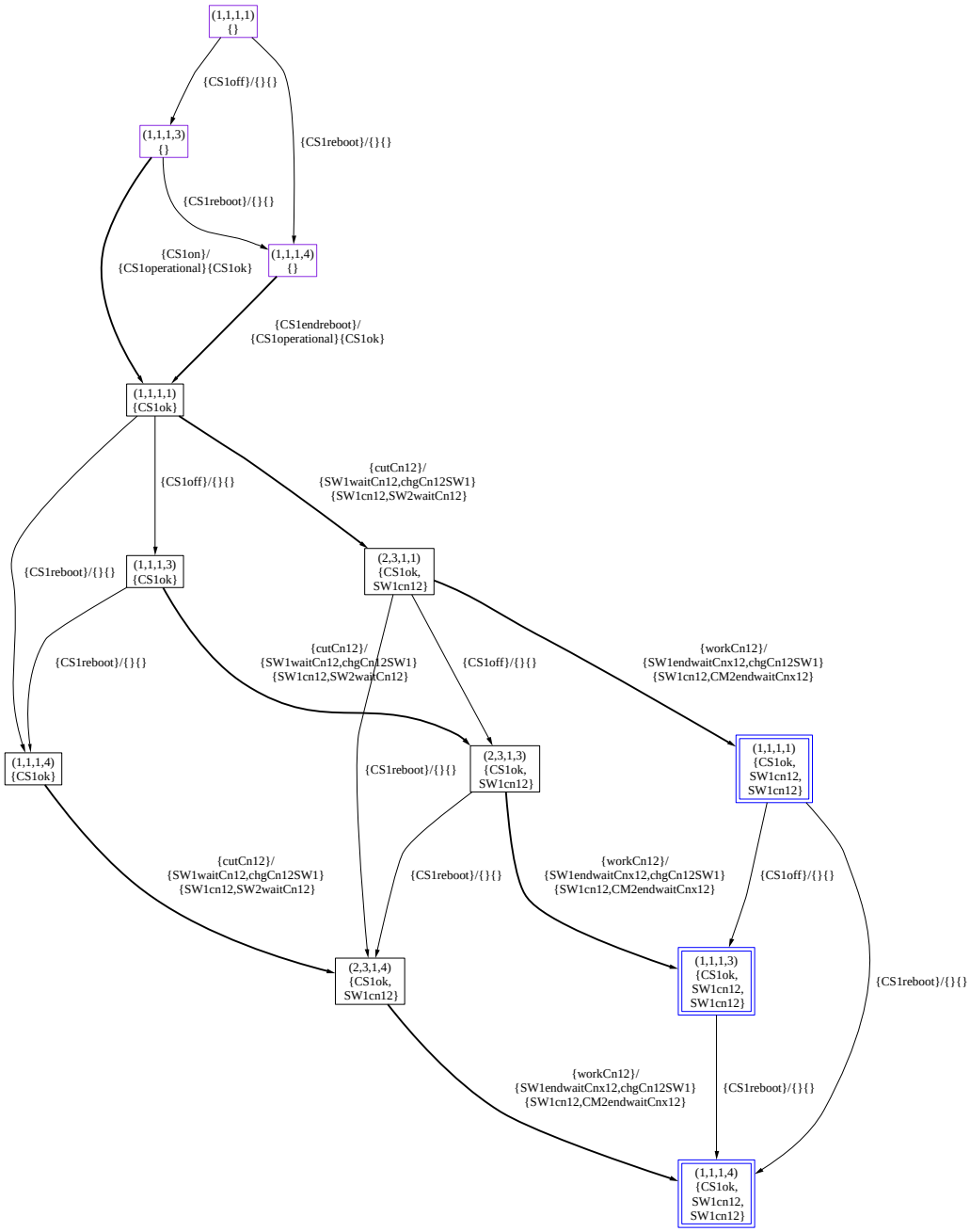

Fig. 11. Subsystem diagnosis represented by $\Delta_{\gamma}^{finite}(O_{\gamma})$ where $\gamma = \{Cn12, SW1cn, SW1ctl, CS1\}$ and $O_{\gamma} = \{CS1ok \prec SW1cn12 \prec SW1cn12\}$.

7.1.2 Partial order reduction

In the following, a summary of partial order reduction theory is given. For more details, see [18,16,5]. We will call an *action* a transition label from any behaviour $\|\gamma\|$ and the set of $\|\gamma\|$ actions will be noted A_{γ}. We will also note en_q the set of actions that can be triggered from the state q in $\|\gamma\|$.

Definition 15 (Independence) *Two actions t_1 and t_2 from A_γ are* independent *in $\|\gamma\| = (I, O, Q, E)$ iff $\forall q \in Q$, if $t_1, t_2 \in en_q$*

(1) $t_1 \in en_{q'}$ where $q \xrightarrow{t_2} q' \in E$;
(2) $\exists q', q'', q''' $ such that $q \xrightarrow{t_2} q' \xrightarrow{t_1} q'' \in E \wedge q \xrightarrow{t_1} q''' \xrightarrow{t_2} q'' \in E$).

Intuitively, two actions are independent if the occurrence of one of them does not affect the occurrence of the other one (condition 1). Moreover, the order in which those actions can occur does not change the state after both occurrences (condition 2).

Definition 16 (Dependence relation) *A* dependence relation *D is a reflexive, symmetric, binary relation such that $\forall (t_1, t_2) \in D$, t_1 and t_2 are not independent.*

This relation induces an equivalence relation between finite sequences of actions. Given two finite sequences v, w of actions from $A_\gamma^\star$, v is equivalent to w according to the relation D iff there exists a set of sequences $\{u_0, \ldots, u_n\}$ such that $v = u_0$, $w = u_n$ and $\forall i \in \{0, \ldots, n-1\}, u_i = \overline{u}t_1t_2\widehat{u} \wedge u_{i+1} = \overline{u}t_2t_1\widehat{u}$ where $\overline{u}, \widehat{u} \in A_\gamma^\star$ and $(t_1, t_2) \notin D$.

Example 10 *Given $v = u_0 = t_1t_2t_3t_4t_5t_6$, $w = u_3 = t_2t_1t_3t_5t_6t_4$, and (t_1, t_2), (t_4, t_5), $(t_4, t_6) \notin D$, we have*

$u_0 = t_1t_2t_3t_4t_5t_6$,
$u_1 = t_2t_1t_3t_4t_5t_6$ *(t_1, t_2 permutation),*
$u_2 = t_2t_1t_3t_5t_4t_6$ *(t_4, t_5 permutation),*
$u_3 = t_2t_1t_3t_5t_6t_4$ *(t_4, t_6 permutation),*

so v is equivalent to w according to D.

This equivalence relation can be extended to infinite sequences. Given two infinite sequences v, w from $A_\gamma^\star$, v and w are equivalent iff for any finite prefix sequence v' of v there exists a finite prefix sequence w' of w such that w' is equivalent to v' and vice versa. This extended relation (for the finite and infinite cases) is called the *partially ordered relation*. This relation is noted $\equiv_D$.

Definition 17 (Trace) *Given a dependence relation D, a* trace *is an equivalence class of sequences defined by the relation $\equiv_D$.*

Thus, a trace represents a set of sequences. Each sequence of the class can be obtained from another one by simply swapping the order of adjacent and independent actions. If s is such a sequence, we note by $[s]_D$ the corresponding trace in which s is included.

7.1.3 Reduced representation

The principle of the reduced diagnosis representation is the following. The diagnosis must represent a set of action sequences, so the idea is to only keep one sequence of each trace that must be represented in a given diagnosis. In order to do that, a dependence relation D_γ between transition labels from $\|\gamma\|$ must be defined. This relation must describe what the dependence of two labels is.

Before giving the definition of D_γ, some notations have to be introduced. Given $t \in A_\gamma$, $E_t \triangleq Rcv(t) \cup Emit(t) \cup Int(t)$ is the set of events that occur in γ when t is triggered. Given any subsystem γ' disjoint of γ, $C_{\gamma'}(t) \triangleq \{c_i \in \gamma' | E_t \cap \Sigma^i_{rcv} \neq \emptyset \vee E_t \cap \Sigma^i_{emit} \neq \emptyset\}$ is the set of components that are directly affected by the transition t in γ'.

Definition 18 (Relation D_γ) *Given t_1 and t_2 in A_γ, $(t_1, t_2) \in D_\gamma$ iff one of the following conditions holds:*

(1) $C_{\Gamma\setminus\gamma}(t_1) \neq \emptyset \wedge C_{\Gamma\setminus\gamma}(t_2) \neq \emptyset$;
(2) $C_\gamma(t_1) \cap C_\gamma(t_2) \neq \emptyset$;
(3) $(Obs(t_1) \neq \emptyset \vee C_{\Gamma\setminus\gamma}(t_1) \neq \emptyset) \wedge (Obs(t_2) \neq \emptyset \vee C_{\Gamma\setminus\gamma}(t_2) \neq \emptyset) \wedge (Obs(t_1) \neq Obs(t_2) \vee C_{\Gamma\setminus\gamma}(t_1) \neq \emptyset \vee C_{\Gamma\setminus\gamma}(t_2) \neq \emptyset)$.

Intuitively, the relation D_γ describes the three criteria of dependence between two transition labels t_1 and t_2. Condition 1 says that if t_1 and t_2 can affect components from $\Gamma \setminus \gamma$, they are dependent because they are part of fault propagations unknown in γ. Condition 2 says that t_1 and t_2 are dependent if they affect common components in γ. Condition 3 is about the observability of t_1 and t_2. From a diagnosis point of view, t_1 and t_2 are also dependent, if they are or could be observable (due to future synchronisations with observable transitions from other subsystems) and the set of emitted observations is not the same.

Because the relation D_γ is just a relation based on events, its computation does not depend on the number of states and transitions of γ, so it is efficient. As a consequence, we can detect on-line if the pair of actions (t_1, t_2) belongs to D_γ or not.

Theorem 3 *The relation D_γ is a dependence relation.*

PROOF. By definition, D_γ is symmetric and reflexive. Now, we have to prove that for any $(t_1, t_2) \notin D_\gamma$, t_1 and t_2 are independent (see definition 15). Let q denote a state of $\|\gamma\|$ and t_1, t_2 be two transitions such that $t_1, t_2 \in en_q$.

Condition 1 Suppose that $q \xrightarrow{t_2} q'$ is a transition of $\|\gamma\|$. Because $(t_1, t_2) \notin$

D_γ, it follows that $C_\gamma(t_1) \cap C_\gamma(t_2) = \emptyset$, so t_1 affects components in γ different from the components affected by t_2. If t_2 is triggered from q, the states of the components affected by t_1 are thus unchanged. Therefore, $t_1 \in en_{q'}$.

Condition 2 Because $C_\gamma(t_1) \cap C_\gamma(t_2) = \emptyset$, t_1 affects components in γ different from the components affected by t_2. Thus, the order of activation of t_1 and t_2 does not change the final state. □

Remark 1 *The relation D_γ is not the unique dependence relation. There are more accurate dependence relations. Nevertheless, the advantage of D_γ is the low cost for checking the dependency of two actions: the check is only based on communication events and does not require a deeper and more expensive on-line analysis of the model.*

Given the dependence relation D_γ, the reduced representation of the diagnosis of γ is defined as follows.

Definition 19 (Reduced representation) *The* reduced representation *of the diagnosis $\Delta_\gamma(O_\gamma)$ is a finite state machine $\Delta_\gamma^{red}(O_\gamma) = (I, O, Q', E')$ such that:*

- $\Delta_\gamma^{finite}(O_\gamma) = (I, O, Q, E)$*;*
- $Q' \subseteq Q$ *is the set of states;*
- $E' \subseteq E$ *is the set of transitions such that every trace $[t_1, \ldots, t_m]_{D_\gamma}$ of $\Delta_\gamma^{finite}(O_\gamma)$ from an initial state to a final state is represented by one transition path $q_0 \xrightarrow{t_1} q_1 \ldots q_{m-1} \xrightarrow{t_m} q_m$ in $\Delta_\gamma^{red}(O_\gamma)$.*

Remark 2 *There are several reduced representations of a diagnosis: it is due to the fact that any sequence of a trace is a good candidate for representing the trace.*

The notions of *initial states* and *final states* in the reduced representation are defined in the manner as in the finite representation (see section 7.1.1). The set of final states does not explicitly represent the set of current states of the subsystem as in the finite representation. However, this set of current states is implicitly represented, each final state representing a set of current states of γ that are equivalent according to the dependence relation. This implicit way of representation is also true in the case of the initial states.

Figure 12 presents the reduced representation of the diagnosis of Figure 11. The transition $t_1 = \{CS1off\}/\{\}\{\}$ and $t_2 = \{CS1reboot\}/\{\}\{\}$ are independent from $t_3 = \{cutCn12\}/\{\ldots\}\{\ldots\}$ and $t_4 = \{workCn12\}/\{\ldots\}\{\ldots\}$. We have $(t_1, t_3), (t_2, t_3), (t_1, t_4), (t_2, t_4) \notin D_\gamma$. Each path from an initial state to a final state represents a trace of events.

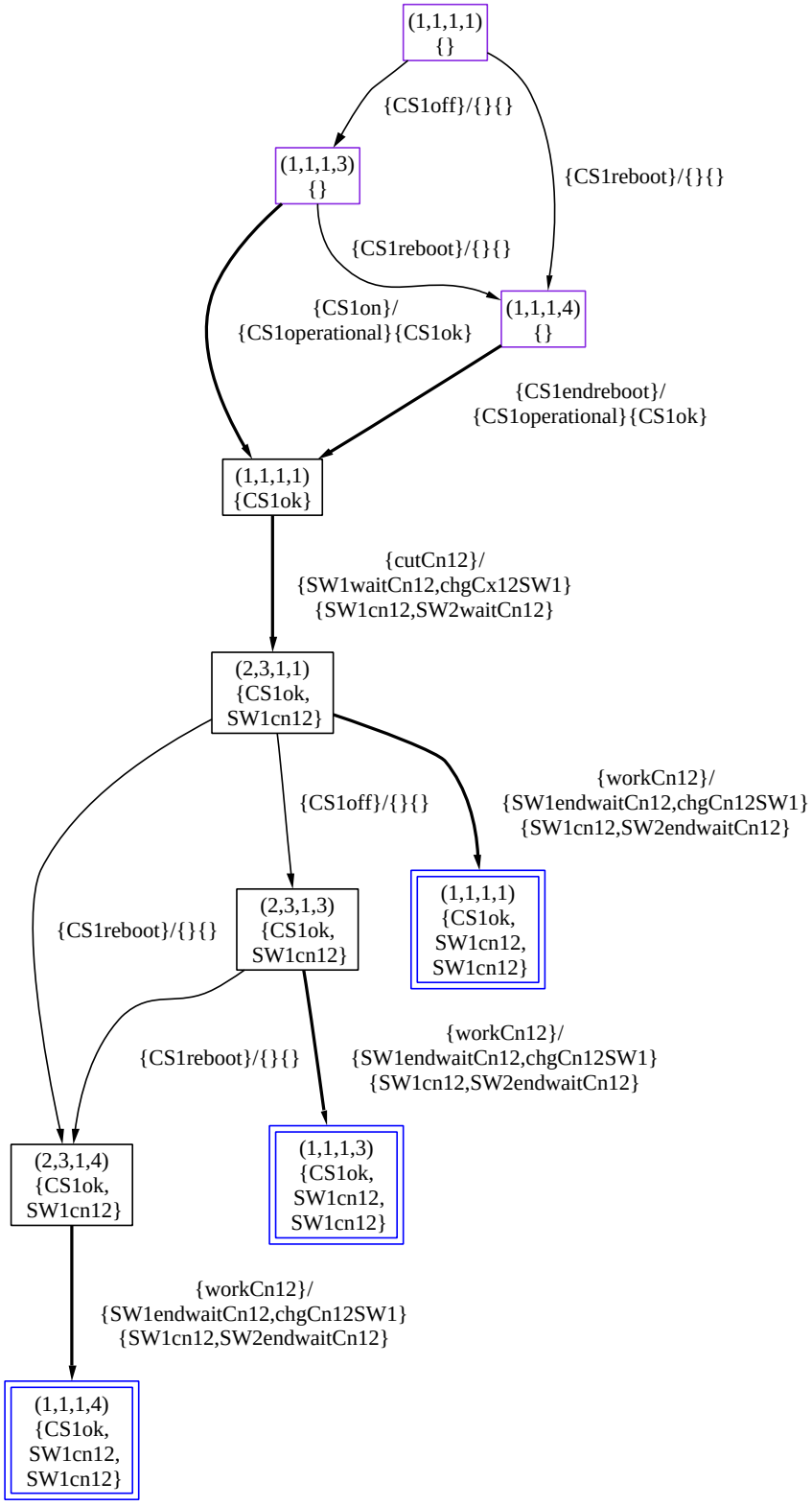

Fig. 12. A reduced representation of the diagnosis from Figure 11.

7.2 Subsystem diagnosis computation

The first step of the decentralised diagnosis approach consists in computing a set of n subsystem diagnoses from the n components c_i. This computation consists in exploring the model Γ_i in order to compute traces that explain the observations O_{c_i}. This exploration is possible because of the tractable size of every Γ_i automaton so that the problem of the subsystem diagnosis computation can be solved by using any centralised diagnosis approach. Moreover, we assume the observation channel between a component and the supervision system is either instantaneous or a bounded FIFO queue (see section 2.2.1), it follows that the computation does not have to take into account the problem of observation overtaking inside the channel. Depending on the system, some components may not be observable at all, in that case the subsystem diagnosis is isomorphic to the model of the component itself and no computation is needed.

In order to be efficient, it is possible to use a *diagnoser* approach [25]. This data structure is a transition system where each transition is labelled with observations and each state contains a pre-compilation of diagnosis information so that it improves the diagnosis computation on-line. More details about this approach can be found in [19][21].

7.3 Merging algorithm

This section presents the merging operation between two diagnoses $\Delta^{red}_{\gamma_1}(O_{\gamma_1})$ and $\Delta^{red}_{\gamma_2}(O_{\gamma_2})$ in order to compute $\Delta^{red}_{\gamma_1\cup\gamma_2}(O_{\gamma_1\cup\gamma_2})$ (see algorithm 1). The proposed algorithm is inspired from the algorithm proposed in [18] which consists in finding runs with a deadlock in a program by checking independences between actions to avoid the state-explosion problem during the search. The proposed algorithm is a decentralised version of the previous algorithm, updated to solve diagnosis problems.

The merging operation has two purposes:

(1) *interaction validation*: events between γ_1 and γ_2 diagnosed by the diagnoses $\Delta^{red}_{\gamma_1}(O_{\gamma_1})$ and $\Delta^{red}_{\gamma_2}(O_{\gamma_2})$ have to be checked;
(2) *reduced diagnosis computation*: the result of the merging operation must be a reduced representation of the diagnosis of $\gamma_1 \cup \gamma_2$.

In order to assure (1), the merging operation must check for any trace (a set of paths) of one diagnosis if this trace can be synchronised with a trace of the other diagnosis (see section 6.2). In order to assure (2), every merged path has to represent a trace of $\|\gamma_1 \cup \gamma_2\|$ according to the dependence relation $D_{\gamma_1\cup\gamma_2}$.

Algorithm 1 (Merging Operation)

1: **Inputs***:* $\Delta^{red}_{\gamma_1}(O_1)$, $\Delta^{red}_{\gamma_2}(O_2)$, $O_{\gamma_1\cup\gamma_2}$
2: **for** *Given* $(q^0_1,\emptyset)\in\Delta^{red}_{\gamma_1}(O_1),(q^0_2,\emptyset)\in\Delta^{red}_{\gamma_2}(O_2)$ *two initial states* **do**
3: $X^0=((q^0_1,q^0_2),\emptyset)$; $sleep(X^0)\leftarrow\emptyset$; $explored(X^0)\leftarrow\emptyset$; $unreliable(X^0)\leftarrow\emptyset$
4: $traces\leftarrow VisitState(X^0)$; $PropagateFixedStates(traces)$
5: **for** $X\in StatesOf(traces)$ **do**
6: **if** $Status(X)\neq fixed$ **then** $Remove(X,traces)$ **end**
7: **end**
8: $\Delta^{red}_{\gamma_1\cup\gamma_2}(O_{\gamma_1\cup\gamma_2})\leftarrow\Delta^{red}_{\gamma_1\cup\gamma_2}(O_{\gamma_1\cup\gamma_2})\cup traces$
9: **end**
10: **Output***:* $\Delta^{red}_{\gamma_1\cup\gamma_2}(O_{\gamma_1\cup\gamma_2})$

11: **Function** $VisitState(X)$
12: $visited(X)\leftarrow$ **true**
13: $trans(X)\leftarrow GiveTransitionsFrom(X)\setminus sleep(X)$
14: **while** $trans(X)\neq\emptyset$ **do**
15: $t\leftarrow Remove(trans(X))$; $explored(X)\leftarrow explored(X)\cup\{t\}$
16: $X'\leftarrow Target(X,t)$
17: $newSleep\leftarrow(sleep(X)\cup explored(X))\setminus(unreliable(X)\cup dependent(t))$
18: **if** $\neg visited(X')$ **then**
19: $explored(X')\leftarrow\emptyset$; $unreliable(X')\leftarrow\emptyset$; $sleep(X')\leftarrow newSleep$
20: $open(X')\leftarrow$ **true**; $paths\leftarrow paths\cup VisitState(X')$
21: **else if** $\exists t\in sleep(X')$ *such that* $t\notin newSleep$ **then**
22: $explored(X')\leftarrow\emptyset$; $unreliable(X')\leftarrow\emptyset$;
23: **if** $\neg open(X')$ **then**
24: $sleep(X')\leftarrow sleep(X')\cap newSleep$
25: $open(X')\leftarrow$ **true**; $paths\leftarrow paths\cup VisitState(X')$
26: **else**
27: $trans(X')\leftarrow trans(X')\setminus sleep(X')$
28: **end**
29: **end**
30: **if** $status(X')\in\{possible,fixed\}\vee open(X')$ **then**
31: $paths\leftarrow paths\cup\{X\xrightarrow{t}X'\}$
32: **if** $open(X')$ **then** $unreliable(X')\leftarrow unreliable(X')\cup\{t\}$ **end**
33: **if** $status(X)\neq fixed$ **then**
34: **if** $status(X')\in\{possible,fixed\}$ **then** $status(X)\leftarrow status(X')$
35: **else** $status(X)\leftarrow possible$ **end**
37 **end**
38: **end**
39: **end**
40: **if** $IsFinal(X)$ **then** $status(X)\leftarrow fixed$ **end**
41: $open(X)\leftarrow$ **false**
42: **return** $paths$

The merging operation assures (2) with the help of the following property.

Proposition 1 *Given γ and γ' two disjoint subsystems, given $t_1, t_2 \in A_\gamma$ two actions of $\|\gamma\|$, if $(t_1, t_2) \notin D_\gamma$ then we have:*

$$\forall t'_1, t'_2 \in A_{\gamma'}, ((t_1, t'_1) \in A_{\gamma\cup\gamma'} \wedge (t_2, t'_2) \in A_{\gamma\cup\gamma'}) \Rightarrow ((t_1, t'_1), (t_2, t'_2)) \notin D_{\gamma\cup\gamma'}.$$

In other words, the defined relation D_γ guarantees that if there are two independent actions t_1 and t_2 in $\|\gamma\|$, then every couple of actions from $\|\gamma \cup \gamma'\|$ based on the actions t_1 and t_2 is also independent. This property is guaranteed by the definition of D_γ (see definition 18). If (t_1, t_2) are not in D_γ, it means that the actions t_1 and t_2 do not interact with actions of γ' at all (they are associated to null events from γ'). So the way t_1 and t_2 can be enabled in the subsystem $\gamma \cup \gamma'$ does not change, they are still independent. On the other hand, some couples (t_1, t_2) of D_γ may be associated to actions t'_1, t'_2 of γ' so that $((t_1, t'_1), (t_2, t'_2))$ may be independent in $\gamma \cup \gamma'$.

Thanks to the proposition 1, the merging operation does not have to retest independent actions computed in $\Delta^{red}_{\gamma_1}(O_{\gamma_1})$ and $\Delta^{red}_{\gamma_2}(O_{\gamma_2})$. It has just to detect new independent actions from the subsystem $\gamma_1 \cup \gamma_2$ to compute $\Delta^{red}_{\gamma_1\cup\gamma_2}(O_{\gamma_1\cup\gamma_2})$.

The algorithm is a depth-first search algorithm based on the search space defined by $\langle \Delta^{red}_{\gamma_1}(O_1), \Delta^{red}_{\gamma_2}(O_2)\rangle$ (see algorithm 1). Each explored state X is built on the fly by synchronising transitions from $\Delta^{red}_{\gamma_1}(O_1)$ and $\Delta^{red}_{\gamma_2}(O_2)$. This synchronisation is done by *GiveTransitionsFrom* (line 13). Formally, given the notations $(q_1, q_2) \stackrel{t=(t_1,t_2)}{\longrightarrow} (q'_1, q'_2)$, the function *GiveTransitionsFrom*(X) where $X = ((q_1, q_2), O_{12})$ is defined as the set of transition labels t such that:

(1) $(q_1, Ind_{\gamma_1}(O_{12})) \stackrel{t_1}{\longrightarrow} (q'_1, O'_1) \in \Delta^{red}_{\gamma_1}(O_1)$ where $Ind_{\gamma_1}(O_{12})$ is the partially ordered set induced from O_{12} which contains all the observations from O_{12} emitted by γ_1;
(2) $(q_2, Ind_{\gamma_2}(O_{12})) \stackrel{t_2}{\longrightarrow} (q'_2, O'_2) \in \Delta^{red}_{\gamma_2}(O_2)$ where $Ind_{\gamma_2}(O_{12})$ is the partially ordered set induced from O_{12} which contains all the observations from O_{12} emitted by γ_2;
(3) $(q_1, q_2) \stackrel{t}{\longrightarrow} (q'_1, q'_2) \in \|\gamma_1 \cup \gamma_2\|$;
(4) there exists $O'^{\gamma_1\cup\gamma_2}_{12}$ such that $Obs_{\gamma_1\cup\gamma_2}(P.(q_1, q_2) \stackrel{t}{\longrightarrow} (q'_1, q'_2)) \diamond O'^{\gamma_1\cup\gamma_2}_{12}$ exists, where P is a transition path from (q^0_1, q^0_2) to (q_1, q_2) in $\|\gamma_1 \cup \gamma_2\|$ and $O'^{\gamma_1\cup\gamma_2}_{12}$ is a subset of $O_{\gamma_1\cup\gamma_2}$ such that $\forall o_1, o_2 \in O'^{\gamma_1\cup\gamma_2}_{12}$, $o_1 \prec o_2$ in $O'^{\gamma_1\cup\gamma_2}_{12}$ iff $o_1 \prec o_2$ in $O_{\gamma_1\cup\gamma_2}$.

Conditions 1 and 2 mean that the transition label t effectively results from the product of a transition from $\Delta^{red}_{\gamma_1}(O_1)$ and a transition from $\Delta^{red}_{\gamma_2}(O_2)$.

Condition 3 means that the result of the synchronisation effectively belongs

to the behaviour of $\gamma_1 \cup \gamma_2$. Condition 4 means that t has an observable behaviour compatible with the observations to explain. If conditions 1, 2, 3 and 4 hold then $(q_1, q_2) \xrightarrow{t} (q'_1, q'_2)$ necessarily belongs to a path of the diagnosis $\Delta_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2})$ and $((q_1, q_2), O_{12}) \xrightarrow{t} ((q'_1, q'_2), O'_{12})$ is a potential candidate for belonging to $\Delta^{red}_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2})$.

Every explored state X has some associated data structures:

- *explored*(X) is the set of actions that have been already explored from the state X;
- *sleep*(X) is the set of actions to avoid when exploring X;
- *unreliable*(X) is the set actions which may be avoided;
- *status*(X) can be *fixed* (X belongs to a path which represents a trace) or *possible* (X may belong to a path which represents a trace, that will depend on the status of the successors of X), *status*(X) is initialised with a value different from *fixed* and *possible*;
- *visited*(X) is true iff X has been or is being explored;
- *open*(X) is true iff X has to be explored.

The computation of the sleep set is based on the independence property of actions. The set *dependent*(t) (line 17) is the set of actions that are dependent of t in $\|\gamma_1 \cup \gamma_2\|$ given the relation $D_{\gamma_1 \cup \gamma_2}$ on these actions. The principle of the algorithm is to explore actions that are not in *sleep*(X) (line 13). An action t in *sleep*(X) is such that t has been already explored from a predecessor X' of X and all actions from X' to X are independent of t. If a trace exists for t from X, this trace has been already computed by exploring t from X' (for more details, see [18]). In line 21, a cycle has been detected during the search, if the old sleep set contains an action which is not in the new one, the state has to be revisited otherwise some traces could be lost.

A state X is fixed if X is guaranteed to belong to a trace of $\Delta^{red}_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2})$. There exist two cases for fixing a state X.

(1) *IsFinal*(X) is true (line 40), then the algorithm has detected a path from X^0 to X which is the representative of one trace. Given $X = (q, O_{12})$ *IsFinal*(X) is defined by:

$$IsFinal(X) \equiv$$

$O_{12} = O_{\gamma_1 \cup \gamma_2} \wedge (\exists q \xrightarrow{t'} q' \in \|\gamma_1 \cup \gamma_2\| \textit{ such that } Obs_{\gamma_1 \cup \gamma_2}(q \xrightarrow{t'} q') \neq \emptyset)$. In other words, X is final if it can explain all the observations and if there exists a behaviour from q in $\|\gamma_1 \cup \gamma_2\|$ which is observable.

(2) A successor of X is fixed (line 34).

The set *paths* contains a set of transitions which represents traces from X to final states of $\Delta^{red}_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2})$. Once X^0 has been visited (line 4), *traces*

contains traces from X^0 to final states of $\Delta^{red}_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2})$. Because of cycles, some states in *traces* are not fixed but only possible. In order to know if one of these states has to be finally fixed or not, one of its successors in *traces* must have a fixed status. *PropagateFixedStates* is in charge of fixing those states (line 4). Some states might stay with a possible status (they belong to cycles which are not part from any traces), they must be eliminated (lines 5-7).

In the following, this merging operation implemented by the algorithm 7.3 is noted $\odot$, so we have:

$$\Delta^{red}_{\gamma_1 \cup \gamma_2}(O_{\gamma_1 \cup \gamma_2}) = \Delta^{red}_{\gamma_1}(O_{\gamma_1}) \odot \Delta^{red}_{\gamma_2}(O_{\gamma_2}).$$

As it is said in Remark 2, there are several reduced representations of a diagnosis. Due to the fact that the $\odot$ is based on a search in a state-space the result can be different depending on the order of merging. Nevertheless, even if the results are different, they are both a reduced representation of the same diagnosis. With $\odot$, we guarantee that the result (a set of traces) is given by using any order of merging the subsystem diagnoses.

7.4 Merging strategy

The merging operation is based on a Cartesian product on subsystem diagnoses. As a consequence, this operation can be very inefficient and has to be used carefully. In order to be as efficient as possible, the idea is to only apply the merging operation when it is necessary. It is the reason why a merging strategy is needed. This merging strategy is based on several criteria defined on the subsystem diagnoses to merge. In this section, we always consider the reduced representation of a diagnosis, but for the sake of clarity, diagnosis notations are simplified: Δ_{γ_i} will refer to $\Delta^{red}_{\gamma_i}(O_{\gamma_i})$.

7.4.1 Definitions

Each subsystem diagnosis Δ_{γ_i} contains traces which claim that the diagnosed components from γ_i have interacted with other components by sending or receiving events belonging to Σ_{int}. We note by $I_{\Delta_{\gamma_i}}(\gamma_j)$ the set of events of γ_i that are supposed to have been sent to or received from the components of γ_j according to the subsystem diagnosis Δ_{γ_i}.

Definition 20 (Inconsistent traces) *A* trace *in a diagnosis* Δ_{γ_i} *is* inconsistent *iff there exists a transition in the trace representative which assumes the emission or the reception of an event* $e \in I_{\Delta_{\gamma_i}}(\gamma_j)$ *such that* $e \notin I_{\Delta_{\gamma_j}}(\gamma_i)$.

A diagnosis Δ_{γ_i} may claim that an event e belongs to $I_{\Delta_{\gamma_i}}(\gamma_j)$ whereas Δ_{γ_j} may claim that e is not in $I_{\Delta_{\gamma_j}}(\gamma_i)$. Traces of Δ_{γ_i} that claim the occurrence of e are said to be *inconsistent*: they will not participate to the global diagnosis and can be immediately discarded from the diagnosis.

In the following, we call *purged diagnosis* of γ_i, the set of traces of Δ_{γ_i} from which inconsistent traces have been eliminated and note it by Δ'_{γ_i}.

Definition 21 (Matchable subsystems) *γ_i and γ_j are* matchable subsystems *iff*

$$I_{\Delta_{\gamma_i}}(\gamma_j) \cap I_{\Delta_{\gamma_j}}(\gamma_i) \neq \emptyset.$$

From this definition, it can be deduced that γ_i and γ_j are matchable iff their purged diagnoses are such that $I_{\Delta'_{\gamma_i}}(\gamma_j) = I_{\Delta'_{\gamma_j}}(\gamma_i) = I_{\Delta_{\gamma_i}}(\gamma_j) \cap I_{\Delta_{\gamma_j}}(\gamma_i) \neq \emptyset$. In the following, we will say that γ_i and γ_j are *k-matchable* subsystems iff they are matchable and $\mid I_{\Delta'_{\gamma_i}}(\gamma_j) \mid = \mid I_{\Delta'_{\gamma_j}}(\gamma_i) \mid = \mid I_{\Delta_{\gamma_i}}(\gamma_j) \cap I_{\Delta_{\gamma_j}}(\gamma_i) \mid = k$.

The purpose of the strategy is to detect and compute subsystem diagnoses that claim no interaction with other subsystems.

Definition 22 (Independent diagnosis) *A subsystem diagnosis Δ_γ is* independent *iff $I_{\Delta_\gamma}(\Gamma \setminus \gamma) = \emptyset$.*

In an *independent* diagnosis, every trace is a complete explanation of the observations of the consider subsystem: for the set of observations of the system, it is impossible to find an explanation where the subsystem has interacted with other subsystems. Every trace of an independent diagnosis shows that any observation from another disjoint subsystem is not caused by a reaction of this subsystem. If an independent diagnosis is detected then it is useless to perform any merging operation on it, the set of interactions to check being empty. Therefore, the global diagnosis is totally and easily represented by a set of independent diagnoses, each diagnosis based on a subsystem disjoined from the others. In the worst case, there is only one independent diagnosis: the global diagnosis.

7.4.2 Principles and algorithm

To improve the efficiency of the merging operation, we apply the two following principles (see algorithm 2) based on the previous definitions.

(1) Detecting and eliminating inconsistent traces. Inconsistent traces uselessly increase the cost of the merging operation. The first principle consists in detecting and eliminating them before performing any merging operation (lines 5-12). Given a diagnosis Δ_{γ_i} of the current diagnoses set, the events

that are sources of inconsistent traces are determined and the elimination is then performed in order to finally obtain the purged diagnosis Δ'_{γ_i}.

(2) Giving priority to the most matchable subsystems. The merging of two diagnoses allows to eliminate some traces by checking the interactions which are claimed by them. Thus, merging two diagnoses which do not claim any interaction between their respective components is not really interesting: the second principle consists in avoiding this useless computation and in giving priority to the most matchable subsystems. Thus, the second stage of the algorithm consists of the computation of sets $kInter(\gamma_i)$ (lines 14-17). Each element of $kInter(\gamma_i)$ is a couple $(\gamma_j, |I_{\Delta'_{\gamma_i}}(\gamma_j)|)$ meaning that γ_i and γ_j are *k-matchable* where $k = |I_{\Delta'_{\gamma_i}}(\gamma_j)|$. The merging strategy then builds partition of diagnoses (with *ChoosePartition* line 20) such that:

- a part has two diagnoses at the most;
- selected diagnoses are such that the set of exchanged events claimed by those diagnoses is as big as possible.

Once a partition of diagnoses is chosen, the diagnoses of each element of the partition (only elements which contain two diagnoses) are merged (line 21), this operation can be done in a parallel way. A new set of diagnoses is obtained where one diagnosis is associated to each element of the partition. The set of possible exchanged events is updated according to the new diagnoses set. Then, the algorithm iteratively proceeds by eliminating new inconsistent traces in the new diagnoses set and then by building the best new partition of diagnoses and merging it. The last stage of the algorithm produces a set of independent diagnoses. The traces of every diagnosis of the resulted set participate to a global trace. In other words, the global diagnosis can be explicitly built by applying new merging operations on this set, but every trace from any independent diagnosis is synchronisable with any trace from any other independent diagnosis.

8 Incrementality

In on-line diagnosis approaches, the purpose is to follow the observed behaviour and to provide a diagnosis as often as possible. Given a time t_1 when a diagnosis has been provided, it is interesting to take into account this diagnosis in order to provide another diagnosis at time t_2 ($t_1 < t_2$), given a new set of observations that occur between t_1 and t_2. This section focuses on this topic. The idea is to propose an *incremental* diagnosis algorithm by extending and updating the diagnosis of time t_1 in order to compute the diagnosis of time t_2 as efficiently as possible. This problematic is called *incremental diagnosis* (see [20]).

Algorithm 2 (Merging strategy)

1: **Input:** *Decentralised model* $\{\Gamma_1, \ldots, \Gamma_n\}$ *of the system* $\{c_1, \ldots, c_n\}$
2: **Input:** *Subsystem diagnoses* $\{\Delta_{c_i}, i \in \{1, \ldots, n\}\}$
3: $D \leftarrow \{\Delta_{c_i}, i \in \{1, \ldots, n\}\}$
4: **do** $\{$*We note* $D = \{\Delta_{\gamma_1}, \ldots, \Delta_{\gamma_l}\}\}$
5: **1-Inconsistent trace elimination**
6: **for** $i \in \{1, \ldots, l\}$ **do**
7: $\Delta'_{\gamma_i} \leftarrow \Delta_{\gamma_i}$
8: **for** $j \in \{1, \ldots, l\}, j \neq i$ **do**
9: $\Delta'_{\gamma_i} \leftarrow ElimInconsTraces(\Delta'_{\gamma_i}, I_{\Delta_{\gamma_i}}(\gamma_j) \setminus I_{\Delta_{\gamma_j}}(\gamma_i))$
10: **end**
11: $Replace(D, \Delta_{\gamma_i}, \Delta'_{\gamma_i})$
12: **end**
13: **2 - Looking for matchable subsystems**
14: **for** $i \in \{1, \ldots, l\}$ **do**
15: $kInter(\gamma_i) \leftarrow \{(\gamma_j, |I_{\Delta'_{\gamma_i}}(\gamma_j)|), I_{\Delta'_{\gamma_i}}(\gamma_j) \neq \emptyset\}$
16: **end**
17: $M \leftarrow \{kInter(\gamma_i), i \in \{1, \ldots, l\}\}$
18: **if** $M \neq \emptyset$ **then**
19: **3 - Applying the merging operation**
20: $\pi_D \leftarrow ChoosePartition(D, M)$
21: $D \leftarrow \{\Delta'_{\gamma_i} \odot \Delta'_{\gamma_j}, \{\Delta'_{\gamma_i}, \Delta'_{\gamma_j}\} \in \pi_D\} \cup \{\Delta'_{\gamma_i}, \{\Delta'_{\gamma_i}\} \in \pi_D\}$
22: **end**
23: **while** $M \neq \emptyset$
24: $\{$ *The set D is a set of independent diagnoses* $\}$
25: **Output***: D*

8.1 Principles and difficulties

Incremental diagnosis is based on two basic concepts.

Definition 23 (Breakpoint) *A* breakpoint t_j *is a date from the supervision system clock.*

Definition 24 (Temporal window) *A* temporal window W_j *is the delay between two consecutive breakpoints* t_j *and* t_{j+1}*.*

The flow of observations belongs to a set of consecutive temporal windows $W_j, j \in \{1, \ldots, m\}$ (see Figure 13). Given a temporal window W_j, the set of received observations in W_j is noted in the following O^{W_j} and the set of observations received before W_j is noted O^{j-1}. The incremental diagnosis is then the problem of computing the diagnosis Δ^j explaining the observations

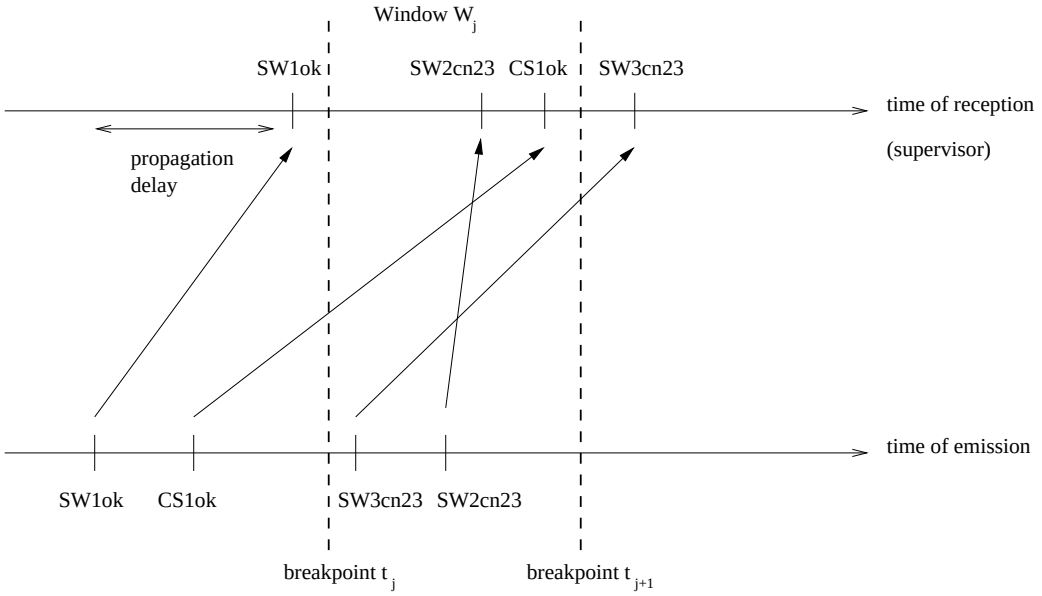

Fig. 13. Temporal windows and breakpoints

O^j given the diagnosis Δ^{j-1} (which explains O^{j-1}) and the observations O^{W_j}.

As already mentioned, the supervision system receives a flow of observations during one temporal window W_j. The problem is that there are some delays between the emission of the messages in the observation channels and their receptions in the supervision system (see Figure 13). Consequently, some messages emitted during W_j may not be received during W_j. Another more problematic consequence is that at the end of W_j there is no guarantee that the supervision system has received enough observations to make a diagnosis. In fact, some messages may not have been received whereas they have been emitted before other received messages; this situation is possible if the unreceived messages are conveyed by observation channels with important delays of transmission.

The choice of the temporal windows is, therefore, fundamental. The update of a diagnosis Δ^{j-1} strongly depends on the nature of the chosen temporal window W_j. In the following subsections, two incremental algorithms are discussed, based on some properties about the chosen temporal windows.

8.2 Sound temporal windows

In this approach, the solution consists in choosing *sound* temporal windows.

Definition 25 (Sound window) *A breakpoint t_j is* sound *iff every message emitted before t_j is received before t_j. A temporal window W_j is* sound *iff t_j and t_{j+1} are sound.*

A sound breakpoint is interesting because it guarantees that the set of messages emitted before this point is effectively received by the supervisor (see Figure 14). In other words, a sound breakpoint guarantees that $\forall o \in O^{j-1}, \forall o' \in O^j \setminus O^{j-1}, o \prec o'$. As a consequence, any update of the diagnosis Δ^{j-1}, taking into account the observations O^j, is only based on its final states.

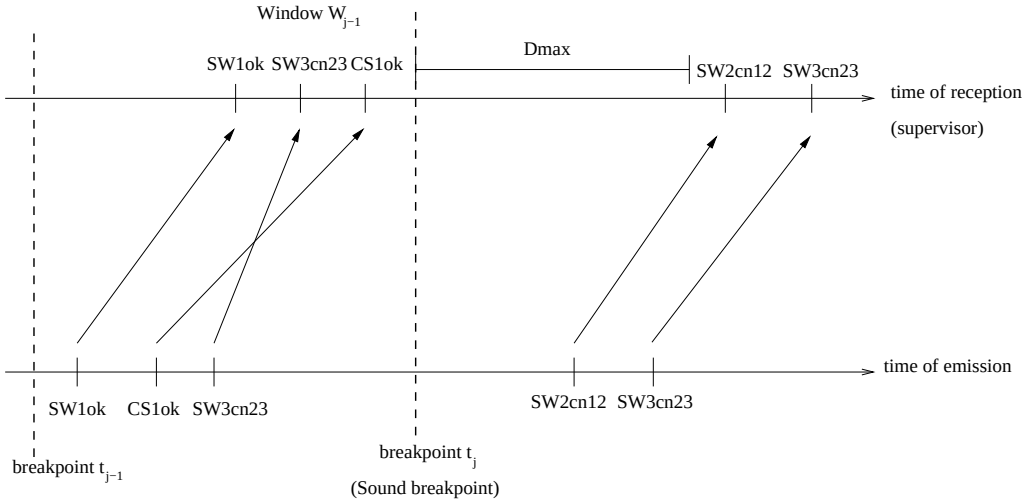

Fig. 14. Sound breakpoint

A sound breakpoint can be detected by taking into account the properties of the observation channels and the date of reception of an observation by the supervisor. For instance, knowing a maximum delay D_{max} of message propagation inside the observation channels, a sound breakpoint t_j is detected if the first observation after this breakpoint is received by the supervision system at time t such that $t > t_j + D_{max}$ (see Figure 14).

In the following, an algorithm which computes the update of the diagnosis is presented (see Algorithm 3).

Algorithm 3 (Incremental diagnosis on sound temporal windows)

1: **Input:** $BS^{W_{j-1}}$
2: **Input:** O^{W_j}
3: **for** $i \in \{1, \ldots, n\}$ **do**
4: $\quad initial(c_i) \leftarrow ExtractStates(BS^{W_{j-1}}, c_i)$
5: **end**
6: $\{\Delta_{\gamma_1}, \ldots, \Delta_{\gamma_p}\} \leftarrow ApplyMergingStrategy(\Delta_{c_1}(initial(c_1), O_{c_1}^{W_j}), \ldots,$
$\Delta_{c_n}(initial(c_n), O_{c_n}^{W_j}), BS^{W_{j-1}})$
7: **Output**: $\Delta^{W_j} = \{\Delta_{\gamma_1}, \ldots, \Delta_{\gamma_p}\}$
8: **Output**: $BS^{W_j} \leftarrow FinalStates(\Delta_{\gamma_1}, \ldots, \Delta_{\gamma_p}, BS^{W_{j-1}})$

The notation BS^{W_j} represents the belief state of the system. [5] . Such a belief state represents the set of global states in which the system could be after the observations O^{W_j}. Given $BS^{W_{j-1}}$, a set $initial(c_k)$ corresponding to the possible initial states of the component c_k at breakpoint t_j is computed by extracting them from $BS^{W_{j-1}}$. Then, the diagnosis of c_k explaining the observations emitted by c_k and received during W_j (noted $O_{c_k}^{W_j}$) from the states $initial(c_k)$ is computed (this diagnosis is noted $\Delta_{c_k}(initial(c_k), O_{c_k}^{W_j})$). Then the merging operation is applied and a set of independent diagnoses is computed. This merging operation is applied according to the strategy defined in section 7.4.2 and depends on the current belief state $BS^{W_{j-1}}$. Once the diagnosis Δ^{W_j} is computed, all the observations O^j are explained and we can extract from the new set of diagnoses the new belief state BS^{W_j} which will be used for the next temporal window.

Once the diagnosis Δ^{W_j} is computed, all the observations O^j are explained. Nevertheless, the diagnosis Δ^j is not totally computed. The explanation of O^{W_j} may have invalidated some traces in $\Delta^{W_{j-1}}$ (it may be impossible to find an explanation of the new observations from a given final state of $\Delta^{W_{j-1}}$) and therefore in Δ^{j-1}. In order to explicitly obtain Δ^j, given Δ^{j-1} and Δ^{W_j}, we must eliminate from Δ^{j-1} all the traces that have no future in Δ^{W_j} and append Δ^{W_j} to the new Δ^{j-1} (see [20] for more details). This operation is made by a refinement operation noted $\oplus$. So we have:

$$\Delta^j = \Delta^{j-1} \oplus \Delta^{W_j}.$$

From a practical point of view, Δ^{W_j} is the interesting information in a context of monitoring; the supervising agent wants to know what have just happened in the system. Thus, applying the refinement operation during the on-line diagnosis is not necessary. Refinement operation is only required when a deeper analysis of the diagnosis is performed, so this operation can be applied off-line (see section 5.4 page 21).

8.3 General case

In the treated example which derives from a real application (see section 9), the hypothesis of sound windows can be applied without loss of generality. In this application, the alarms are instantaneously emitted and received when a problem occur. A problem causes a large packet of alarms at a given time, so it is easy to define a sound temporal window that surrounds this packet. So the algorithm previously defined can be used to solve the problem. Nevertheless,

[5] The computation of the belief state is not the topic of this paper. For efficiency purposes, its representation is symbolic and uses binary decision diagrams [4]

in theory, sound windows may not exist and this section describes this general case where the given temporal windows are arbitrarily chosen (see [20] for more details).

The purpose of the incremental diagnosis is to always provide a diagnosis for a given temporal window, thus the method in the general case must take into account two kinds of observations:

(1) the observations that have effectively been received;
(2) the observations that have been emitted but are not received yet.

If the algorithm 3 is used on an unsound temporal window, some explanations in the resulting diagnosis may be missing. In fact, the merging operation assumes that every emission of message in the observation channels is effectively received, so explanations that require the emission of observable events which have not been received yet, are not computed. In the worst case, it is possible that the only explanations of a given set of observations are based on the fact that messages are still in the observation channels, in that case, the algorithm 3 is unable to provide any explanation. In the general case, in order to provide a diagnosis for each temporal window, the unreceived messages must be *guessed*, some observations have to be supposed *uncertain* [12]. In this context, a new diagnosis structure must be defined that allows to represent traces under the hypothesis of emitted but unreceived observations. This structure is called *extended diagnosis*. Basically, an *extended diagnosis* has the same representation of a *diagnosis* except that an extended diagnosis state X is formed as a couple (q, O) where O may contain unreceived observations that are also explained. The set of final states of an extended diagnosis, which corresponds to the extended belief state $BS_{ext}^{W_j}$, is composed of states (q, O) where $O_{W_j} \subseteq O$. The computation of an extended diagnosis needs the following hypothesis.

Hypothesis 5 *Every observation channel from a component to the supervision system is bounded by a known number "*capacity*".*

In fact, if the size of the channel was unknown, an extended diagnosis would have to guess an unknown number of unreceived observations.

The incremental diagnosis algorithm in the case of unsound temporal window is obtained by replacing in the algorithm 3 the use of diagnoses (resp. belief states) by the use of extended diagnoses (resp. extended belief states). There are two main differences. One difference is on the computation of the $initial(c_k)$ sets (line 4). Instead of extracting states of c_k from every state of the previous extended belief state, we have to only extract them from a subset of it: the interesting states are only the ones which are compatible with the new set of received observations (observations that have been supposed to be received and that have been effectively received). The second difference is on the computation of the extended diagnoses of components (line 6). For some

c_k, some observations of $O_{c_k}^{W_j}$ have been already explained in the previous temporal window, it follows that it is not necessary to do it again, secondly, some assumptions must have been made about possible unreceived observations at the end of W_j in order to achieve the extended diagnosis computation.

As far as the refinement operation is concerned, the operator is the same and we have:

$$\Delta_{ext}^{j} = \Delta_{ext}^{j-1} \oplus \Delta_{ext}^{W_j}.$$

8.4 Relation between diagnosis and extended diagnosis

With the help of the extended diagnosis notion, it is still possible to provide a diagnosis for any temporal window. The extended diagnosis also assures that no explanation is missing. In fact, by definition, an extended diagnosis represents a set of explanations that explain not only the received observations but also a set of possible unreceived observations. If there is an explanation of the received observations which does not require assumptions about unreceived observations, then this explanation is contained in the extended diagnosis. As a consequence, it can be easily shown that, for any breakpoint t_j, we have:

$$\Delta^j \subseteq \Delta_{ext}^j.$$

Moreover, if we have the guarantee that, after a given temporal window W_j, the breakpoint t_{j+1} is sound then no assumption about unreceived events is required. In that case, it can be shown that:

$$\Delta^j = \Delta_{ext}^j.$$

This is especially the case when t_{j+1} is the date when the supervised system stops working.

9 Experimental results

For testing the approach above, we have used a model of a telecommunication network coming from the project MAGDA. This model is based on a real SDH network (*Synchronous Data Hierarchy*) and it has been defined during the collaboration between academic and industrial partners of the MAGDA project.

The studied network is a ring of 4 ADM multiplexers (ADM: *Add and Drop Multiplexer*) (see Figure 15). Each multiplexer is located in a different town of the area *Ile de France*: Aubervilliers, Gentilly, Montrouge, and St Ouen. Whereas the Aubervilliers ADM only transmits data from Gentilly to St Ouen and vice versa, the other ADMs have connections with clients of the network via *PDH* and *STM1* connections.

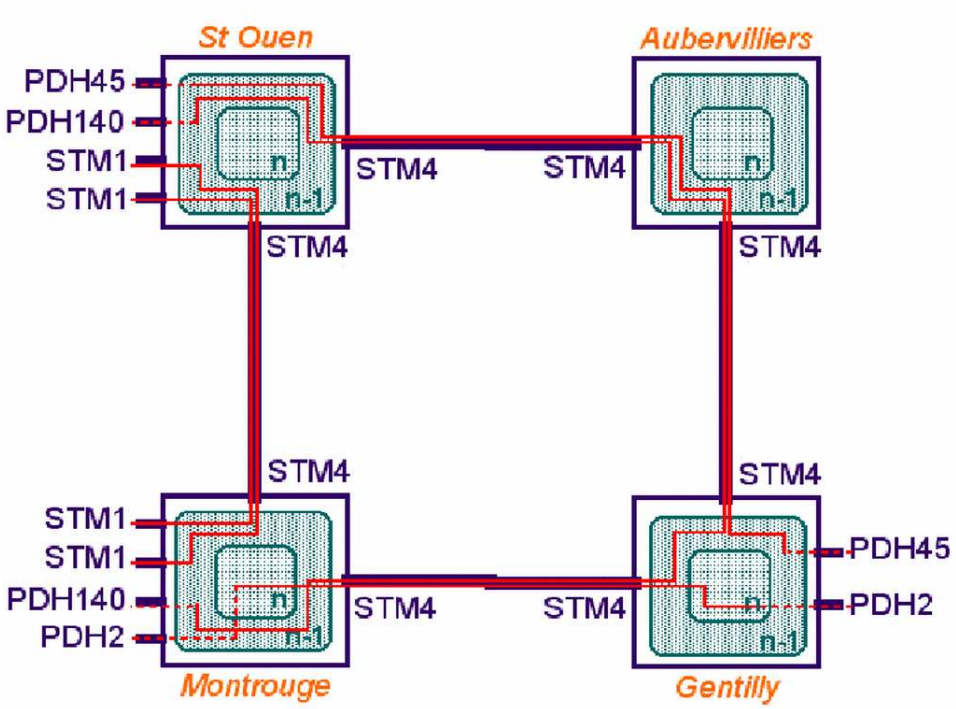

Fig. 15. Topology of the SDH network: a ring of 4 ADMs

This network is managed with the help of *managed objects* which are defined in the SDH norms. Each object corresponds to a functionality of a part of a multiplexer. Figure 16 presents the 23 managed objects associated to the multiplexer of Montrouge. These objects take into account that the SDH protocol is hierarchical (from SPI (*Synchronous Physical Interface*) to LOP (*Low Order Path*)). Globally, the network is composed of 72 managed objects, each object behaviour is modelled by a communicating automaton; the global centralised model, if it was explicitly built by the free product of these automata, would have about 5.6×10^{47} states.

Each managed object can emit some alarms if it detects a problem. It can also emit alarms if it receives messages from other managed objects. In particular, if an object from a ADM x detects a problem, it emits a message to the same objects from the other ADMs y and z where y and z are the neighbours of x. Figure 17 shows the model which describes the au3CTP component on the

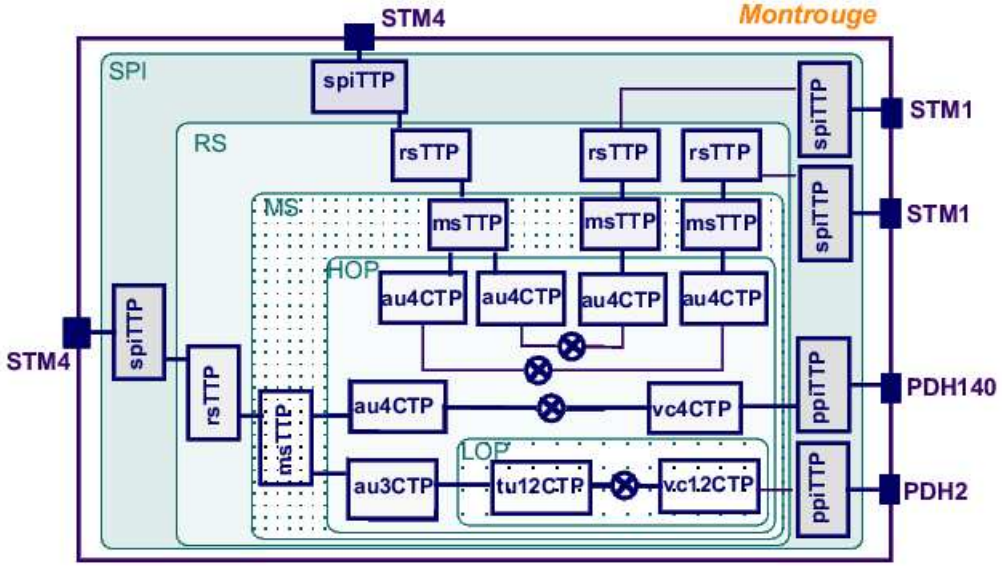

Fig. 16. Montrouge Add and Drop Multiplexer

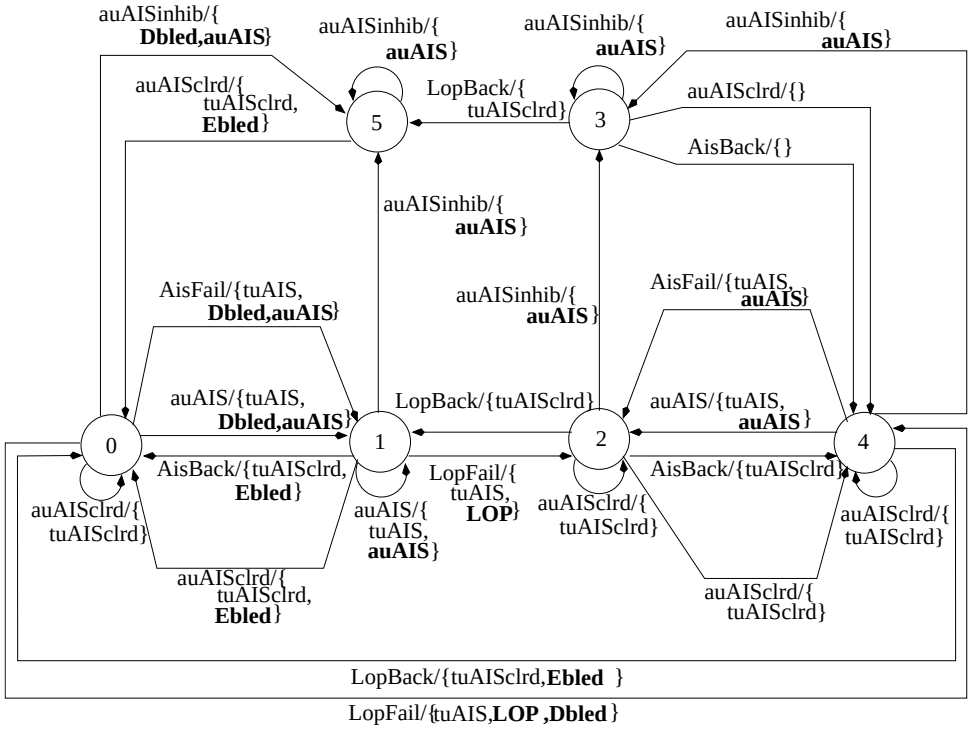

Fig. 17. Montrouge: model of the component au3CTP

Montrouge site. The problems that can occur on this component are modelled by *AisFail, AisBack* (problem of alarm indication signal), *LopFail, LopBack* (loss of pointer). Such problems can also occur on other sites, the component detects those problems by the reception of messages from the *msTTP* neighbour such as *auAIS,auAISclrd, auAISinhib*. This information is propagated to *tu12CTP* with the help of events like *tuAIS, tuAISclrd*. The observations emitted by this component (in bold) are *Dbled* (Disabled), *Ebled* (Enabled), *auAIS* and *LOP*.

9.2 Results

During the MAGDA project, 8 scenarios of faults have been defined. A scenario consists of a set of failures that occur in the network and the set of alarms that are observed in reaction of these failures. These scenarios have been defined in order to characterise typical faulty situations, in particular, some scenarios contain with multiple faults and masking phenomena (scenarios *S5* and *S8*). We have experimented our approach by diagnosing each scenario from each set of alarms using our diagnostic tool *DDyp* [21]. For these experiments, we have considered that the set of observed alarms corresponds to a sound temporal window.

9.2.1 Results on the merging strategy

Table 1 presents the time of computation needed for diagnosing the different scenarios if we use a unique machine (Pentium III 1Ghz) (computation of the subsystem diagnoses and the global diagnosis, no parallel computation has been used in this experiments for better comparisons, see [21] for details on parallel computations). Based on the set of received alarms, for each scenario, our approach finds the failures defined in the scenario but also it finds out other failure scenarios that can explain the same observations.

In order to show that the strategy we propose is fundamental in a decentralised approach, Table 1 presents a comparison between the results from 4 different strategies. The first strategy is the one computed with the help of the algorithm 2. The second strategy is the same as the first one, except that the merging order is such that two non-matchable diagnoses are merged. The third strategy consists in merging like in strategy 1 but without eliminating incompatible traces before the merging. The fourth strategy is like the strategy 2 but without eliminating incompatible traces before the merging.

Strategy 1 shows that the on-line diagnosis computation is possible on the SDH network when we are dealing with typical diagnosis situation (single and multiple failures). Strategy 2 shows the choice in the merging ordering

Scenarios	Observed alarms	Strategy 1	Strategy 2	Strategy 3	Strategy 4
S1: Laser failure (St Ouen)	24	3s 590ms	4s 200ms	16s 540ms	>5mn
S2: AU3 failure (Aubervilliers)	4	1s 300ms	1s 300ms	1mn 53s	>5mn
S3: Laser failure (Gentilly)	26	1s 780ms	1s 910ms	>5mn	>5mn
S4: RS failure (Aubervilliers)	14	1s 600ms	2s 30ms	49s	>5mn
S5: Multiple failures (S3 with S4)	36	2s 620ms	5s 500ms	5s 430ms	3mn 45s
S6: BER failure (Aubervilliers)	11	1s 780ms	2s 320ms	24s 240ms	57s 440ms
S7: RS failure (Gentilly)	14	1s 480ms	1s 700ms	2mn 55s	>5mn
S8: Multiple failures (S6 and S7)	21	1s 830ms	3s 90ms	3s 30ms	>5mn

Table 1
Diagnosed scenarios with different strategies of merging

is important and has to take into account interactions between subsystem diagnoses. For better comparisons, strategy 2 merges the same diagnoses as strategy 1 but in a different order. Therefore, strategy 2 is defined according to strategy 1. In practice, if we do not care about interactions at all, such a strategy is unable to determine independent diagnoses so the time of computation can strongly increase because the result of the strategy will be only one diagnosis. Strategy 3 shows the elimination of the traces *a priori* has a big impact on the performance of the merging operation. Strategy 4 is very time-consuming. The merging of diagnoses that are not matchable corresponds to the Cartesian product. Moreover, incompatible traces are uselessly computed and then invalidated by the merging with another diagnoses in future steps. Strategy 4 shows that trajectory elimination and good ordering strategy have cumulative and benefit effects on the merging operation.

9.2.2 Characteristics of the computed diagnoses

Table 2 presents some characteristics of the computed diagnoses by the strategy 1 for the scenarios defined in Table 1. The characteristics are the following.

- *Involved comps. max*: the maximal number of components that are involved in a fault propagation.
- *Indep. diagnoses*: the number of independent diagnoses.
- *Red. states max*: the number of states in the biggest independent reduced diagnosis.
- *Red. trans. max*: the number of transitions in the biggest independent reduced diagnosis.
- *States max*: the number of states in the biggest independent unreduced diagnosis followed by the reduction rate.
- *Trans. max*: the number of transitions in the biggest independent unreduced diagnosis followed by the reduction rate.
- *Strategy 1 overhead*: the overhead time needed by Strategy 1 to compute the diagnoses without any reduction.

Scenarios	Involved comps. max	Indep. diagnoses	Red. states max	Red. trans. max	States max	Trans. max	Strategy 1 overhead
S1	36	26	81	82	210 / 61%	609 / 87%	34%
S2	23	28	3	2	3 / 0%	2 / 0%	0%
S3	28	27	10	10	38 / 74%	81 / 63%	7%
S4	40	29	15	14	35 / 57%	62 / 77%	18%
S5	40	25	36	36	190 / 81%	515 / 93%	76%
S6	36	27	19	18	29 / 34%	45 / 60%	0%
S7	28	27	6	5	9 / 33%	11 / 54%	1%
S8	36	25	17	16	35 / 51%	53 / 70%	8%

Table 2
Diagnosis characteristics: comparison of the reduced/unreduced representations.

The first significant result of Table 2 is the fact that for each analysed scenario our diagnosis system was able to detect several independent diagnoses. This is due to the fact that the propagation of a failure does not generally involve the whole system but only a subpart of it (the biggest involved subsystem is composed of 40 components over 72 in scenario 5), the other parts behaving independently from the occurrence of the diagnosed failures.

The second result presented in Table 2 is about the reduction of the diagnoses. Strategy 1 has been applied to merge diagnoses without any reduction techniques in order to analyse the impact of the reduction of the diagnoses. The reduction percentages are important and show that the set of components of the system have a high degree of concurrency (independent behaviours) that are detected by our dependence relation. Moreover, the merging of the reduced diagnoses is more efficient than the merging of the non reduced ones.

The main reasons are that, firstly, the merge of two unreduced diagnoses needs to explore a bigger state-space and, secondly, the computation of the dependence relation is efficient enough. It results that the overhead for reducing is neglectable.

9.3 Complexity discussion

Results about the efficiency merging operation have been presented relying on a real and complex large system. The merging operation is efficient in this example because it benefits from different approaches that are combined and well-suited. In this section, an informal discussion about the complexity of the merging operation is presented.

Firstly, the merging operation is based on the *divide and conquer* paradigm and exploits its efficiency. In this case, this paradigm is efficient because the set of behaviours diagnosed locally is usually very small compared to the number of possible local behaviours. This fact is true if the diagnosed system has good observability properties (lot of different observation types, very few unobservable components) and the temporal windows are small enough. Moreover, with good observability properties in the system, we expect that the global diagnosis is exponentially smaller that the global behaviour.

The second reason of the merging operation efficiency is the strategy. The purpose of the strategy is to minimise the computations by avoiding the merging of independent diagnoses that is complex (cartesian product) and useless. The representation of the global diagnosis thanks to the set of independent diagnoses is exponentially smaller (relatively to the number of independent diagnoses) than the representation of the global diagnosis as a unique finite-state machine. In large systems, independent diagnoses exist because a failure does not usually propagate its consequences on all the components of the system but on a subpart of it.

The third reason of its efficiency is the use of partial order techniques. Those techniques are well-known in model-checking to exponentially reduce the complexity of a space search algorithm in the good cases. A good case is a system with a lot of independent events in it which is typically the case of the systems we consider. The partial-order techniques are efficient if a trade-off is found between the amount of detected independent events during the search and the complexity of the algorithm to detect these independences. In our case, the independence are detected in a local and incremental manner (during each merging operation) with a detection algorithm which is constant and does not consequently create any overhead in the merging algorithm.

10 Related work

There are several works which propose a framework for the decentralised diagnosis of discrete event systems. In [8], the authors propose a monitoring system based on the fact that the supervised system is observed by a set of sensors. The framework consists of a set of diagnosers [25,26], each diagnoser aiming at explaining the observations from one site. When an observation occurs on a sensor, the corresponding diagnoser updates its diagnosis. Because the diagnoser is computed from a global model of the system, the diagnosis proposed by the diagnoser is a global diagnosis. The merge operation consists here in exchanging messages between diagnosers in order to make a consensus of the diagnosis proposed by each diagnoser. This approach is well-suited for monitoring because the diagnoser approach is efficient. Nevertheless, using this approach for large discrete event systems is not possible due to the impossibility of computing the global model of the system.

The authors of [2] propose the framework for the diagnosis of *active systems*. In this framework, based also on communicating finite state machines and communication channels, the diagnosis computation consists in unfolding the set of automata given the set of observations. The purpose of the approach is to compute the global diagnosis (also called *active space*) as an automaton giving all the explanations. The main difference with our work is about the efficiency of the approach, the active space approach being an off-line technique: the set of observations is thus considered as complete (no incremental diagnosis problem) and the efficiency of the method is not crucial. To compute the global diagnosis, the authors propose a modular reconstruction based on the topology of the system by firstly building subsystem diagnoses and secondly merging the set of diagnoses in a hierarchical manner [3]. The merging strategy we propose is based on the same idea. Nevertheless, the main difference with our merging strategy is that the *reconstruction plan* builds, in any cases, the global diagnosis of the system. No reduction techniques are used to compute an efficient representation of the global diagnosis moreover the modular reconstruction does not manage the fact that some subsystems may have independent diagnosed behaviours and that the merging of them is useless. This work has been extended for integrating synchronous and asynchronous behaviours in the same model in the framework of *polymorphic systems* [14].

Very recently, in [13], a new technique, called *Continuous Diagnosis*, has been proposed in order to extend the active system approach for monitoring purposes. In this approach, the authors propose to mix the diagnoser approach (well-suited for monitoring) and the active space approach (well-suited for diagnosing large scale discrete-event systems). The main idea is to compute on-line, from the model, the state of a finite-state machine called a *monitor* that currently gives the diagnosis of the system (failure localisation). In the

continuous diagnosis approach, the temporal windows only consist of one observation and is supposed to be sound (like in the classical diagnoser approach [25,26]). Given a temporal window, the belief state (augmented with a diagnosis information) is represented by the current state of the monitor. When an observation occurs, the computation consists in searching for observable transitions in the model that match the observation and in computing the unobservable bahaviour that could occur after the reception of the observation (called the *silent closure* in the paper). This approach assumes that the global unobservable behaviour occuring after an observation is computable on-line. Our merging strategy with the help of the partial reduction techniques can contribute to increase the efficiency of the silent closure computation.

Another set of works have also been proposed to monitor stochastic systems. The idea consists in using probability in order to only compute a set of preferred explanations (the most likely explanations generally) among the set of all explanations. As an example, in the Livingstone project [29], the system is modeled as a set of transition systems and a global probability distribution which rules the triggering of the transitions in the modeled system. In this framework, the purpose is to compute a belief state (the set of the most likely states of the system) by monitoring a sequence of observations. This work has been extended in [11] to compute also the most likely behaviours (called *trajectories* in the cited paper). A more recent approach is also proposed in [10]. The framework is based on a set of stochastic diagnoser agents which are in charge of computing the most likely local diagnoses for the set of global observations. An agent only knows about the behaviour of one subsystem. The merging operation is implemented by message exchanges between several agents in order to check diagnosis interactions but also to check if the result is likely or not. In this framework, the idea is to never compute a global diagnosis, the local diagnoses are checked but in order to compute failure propagation, another computation is needed (association of local diagnosis). In our framework, such local diagnoses are obtained by projecting the set of independent diagnoses to the given subsystem. The approaches based on stochastic systems are more efficient (only a subset of the complet diagnosis is computed) but they have also several problems. Firstly, a probability information is neccessary and is difficult to acquire from a real application (the expertise is generally poor and automatic training methods have to be used). Secondly, the most likely explanations of a sequence of observations are not necessary the most interesting ones: the occurrence of a serious failure is generally unlikely. Finally, the monitoring of a system can be very inefficient due to the fact that a likely explanation for one temporal window can become very unlikely in the next window and a backtrack is then necessary.

11 Conclusion and perspectives

In the paper we propose a framework for the on-line diagnosis of large scale discrete event systems. Dealing with large discrete event systems implies that the use of a global model is impossible. The proposed formal framework allows to model large discrete event systems in a modular way. Moreover, thanks to the properties of the synchronisation operation (associativity and commutativity), any decentralised reasoning can be performed on the system, following the *divide and conquer* paradigm.

Given that framework, we then propose an on-line decentralised diagnosis approach. Because the system emits observations from several subsystems, we can divide the diagnosis problem into several diagnosis subproblems based on a set of subsystems. Then, once those diagnoses are established, a merge operation, based on the synchronisation operation, is necessary to obtain the global diagnosis. The purpose of the merge operation is to build the missing information which is in the global model: interaction checking. In order to make an on-line diagnosis, this operation has to be efficient. For this reason, several ideas have been developed in this paper. Firstly, the diagnosis representation has to be efficient. Representation problem efficiency is due to the concurrency of the system. Our proposal is to use partial order reduction techniques to solve the problem. The second point is the proposal of a merging strategy. This strategy dynamically computes an efficient way to merge the diagnoses. The basic idea is to dynamically recognise diagnostic problems that are independant from each other and benefit from this dynamic independence. This recognition does not result from an a priori analysis, as it is usually done in the literature, but from a analysis based on observed interactions which provides more accurate results. Finally, in the context of system monitoring, being efficient also means being incremental. To achieve that, we define the incremental diagnosis problem which takes advantage from the diagnosis previously computed to compute the new diagnosis given a new flow of observations.

This framework has been implemented for the monitoring of telecommunication networks [21] and has been integrated and validated in the context of the MAGDA project. The purpose of this project was to provide a complete supervision chain from the modeling of the system to the ergonomic view of failure propagations to a supervisor. The presented framework proposes a way to model a large discrete event system, to provide, on-line, a complete diagnosis of the system. The diagnosis being exhaustive can be presented in several ways to a supervisor agent depending on his needs (on-line analysis, deep off-line analysis,...) at a given time. The studied network is a real case and the promising results of this study have been reported in this paper.

The perspective of this work are numerous. Firstly, the described framework can be used to solve the diagnosability problem. Previous works on that problem needs the computation of a global model, so there is no known algorithm for dealing with large scale discrete event systems. One challenge is to propose a solution to this problem inside the proposed framework. Another problem to deal with is the reconfiguration of systems. In this problem, not only the diagnosis system has to deal with observations but also with on-line evolution of the system (connection reconfiguration in the case of telecommunication networks). Finally, this framework could be extended to model large scale autonomous systems by mixing diagnosis and planning approaches [29].

A Proof of the theorem 1

Theorem 1 *Let γ_1 and γ_2 be two disjoint subsystems, then*

$$\|\gamma_1 \cup \gamma_2\| = \|\|\gamma_1\|, \|\gamma_2\|\|.$$

PROOF. Let γ_1 and γ_2 denote two disjoint subsystems with $\{\Gamma_{i_1}, \ldots, \Gamma_{i_k}\}$ and $\{\Gamma_{j_1}, \ldots, \Gamma_{j_l}\}$ the respective component sets of γ_1 and γ_2. The behaviour of the subsystem $\gamma_1 \cup \gamma_2$ is a finite state machine $\|\gamma_1 \cup \gamma_2\| = (I, O, Q, E)$ included in the free product

$$\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k}, \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle.$$

$\|\gamma_1\|$ is included in the free product $\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k} \rangle$ and $\|\gamma_2\|$ is included in the free product $\langle \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle$. Therefore, the behaviour of the subsystem composed of the automata $\{\|\gamma_1\|, \|\gamma_2\|\}$ is by definition a finite state machine (I', O', Q', E') included in the free product:

$$\langle \langle \Gamma_{i_1}, \ldots, \Gamma_{i_k} \rangle, \langle \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle \rangle = \langle \Gamma_{i_1}, \ldots, \Gamma_{i_k}, \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle.$$

Consequently, $I = I'$, $O = O'$ and

$$Q, Q' \subseteq \prod_{p \in \{i_1, \ldots, i_k, j_1, \ldots, j_l\}} Q_p.$$

To prove the result, it suffices now to show that $E = E'$.

($E' \subseteq E$) E is the set of synchronised transitions from $\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k}, \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle$. E' is the set of synchronised transitions from $\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k}, \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle$ resulting from the product of transitions from $\|\gamma_1\|$ and $\|\gamma_2\|$. Therefore, E'

is necessarily contained in E.

$(E \subseteq E')$ Every transition T of E is as follows:

$$T = (q_{i_1} \xrightarrow{t_{i_1}} q'_{i_1}, \ldots, q_{i_k} \xrightarrow{t_{i_k}} q'_{i_k}, q_{j_1} \xrightarrow{t_{j_1}} q'_{j_1}, \ldots, q_{j_l} \xrightarrow{t_{j_l}} q'_{j_l})$$

with $(q_{i_1} \xrightarrow{t_{i_1}} q'_{i_1}, \ldots, q_{i_k} \xrightarrow{t_{i_k}} q'_{i_k})$ a transition from $\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k} \rangle$ and $(q_{j_1} \xrightarrow{t_{j_1}} q'_{j_1}, \ldots, q_{j_l} \xrightarrow{t_{j_l}} q'_{j_l})$ a transition from $\langle \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle$. The transition T is synchronised. If the $(t_j)_{j \in \{i_1, \ldots, i_k\}}$ or the $(t_j)_{j \in \{j_1, \ldots, j_l\}}$ are null, then by definition the corresponding transitions from $\langle \Gamma_{i_1}, \ldots, \Gamma_{i_k} \rangle$ and $\langle \Gamma_{j_1}, \ldots, \Gamma_{j_l} \rangle$ are synchronised and T is in E'.

Otherwise, we have $card(\{t_j, j \in \{i_1, \ldots, i_k, j_1, \ldots, j_l\} | rcv(t_j) \in \Sigma_{exo}^{\gamma_1 \cup \gamma_2}\}) \leq 1$. Consequently, we obtain

$$card(\{t_j, j \in \{i_1, \ldots, i_k\} | rcv(t_j) \in \Sigma_{exo}^{\gamma_1}\}) \leq 1,$$

$$card(\{t_j, j \in \{j_1, \ldots, j_l\} | rcv(t_j) \in \Sigma_{exo}^{\gamma_2}\}) \leq 1.$$

For each non null $t_j, j \in \{i_1, \ldots, i_k, j_1, \ldots, j_l\}$, we also have:
(1) $\forall e \in emit(t_j) \cap \Sigma_{int}^{\gamma_1 \cup \gamma_2}, \exists r \in \{i_1, \ldots, i_k, j_1, \ldots, j_l\}, e = rcv(t_r)$;
(2) $\forall rcv(t_j) \in \Sigma_{int}^{\gamma_1 \cup \gamma_2}, \exists r \in \{i_1, \ldots, i_k, j_1, \ldots, j_l\} | rcv(t_j) \in emit(t_r)$.
If $j \in \{i_1, \ldots, i_k\}$, then, from (1), we obtain

$$\forall e \in emit(t_j) \cap \Sigma_{int}^{\gamma_1}, \exists r \in \{i_1, \ldots, i_k, j_1, \ldots, j_l\}, e = rcv(t_r).$$

Because γ_1 and γ_2 are disjoint, $\Sigma_{int}^{\gamma_1}$ does not contain any events from the subsystem γ_2, so $\Sigma_{int}^{\gamma_1} \cap (\bigcup_{r \in \{j_1, \ldots, j_l\}} rcv(t_r)) = \emptyset$. Finally, from (1), we have

$$(\star) \quad \forall e \in emit(t_j) \cap \Sigma_{int}^{\gamma_1}, \exists r \in \{i_1, \ldots, i_k\}, e = rcv(t_r).$$

Using the same way of reasoning, the property (2) implies:

$$(\star\star) \quad \forall rcv(t_j) \in \Sigma_{int}^{\gamma_1}, \exists r \in \{i_1, \ldots, i_k\} | rcv(t_j) \in emit(t_r).$$

Finally, we know that:

$$\exists j \in \{i_1, \ldots, i_k, j_1, \ldots, j_l\} | rcv(t_j) \in \Sigma_{rcv}^{\gamma_1 \cup \gamma_2}.$$

Two cases hold.
(1) If $j \in \{i_1, \ldots, i_k\}$ then $\exists j \in \{i_1, \ldots, i_k\} | rcv(t_j) \in \Sigma_{rcv}^{\gamma_1}$.
(2) If $j \notin \{i_1, \ldots, i_k\}$, suppose for the sake of contradiction that $\forall j \in \{i_1, \ldots, i_k\} | rcv(t_j) \in \Sigma_{int}^{\gamma_1}$. With the help of $(\star)$ and $(\star\star)$, it follows that the transition $(t_j)_{j \in \{i_1, \ldots, i_k\}}$ represents a cyclic instantaneous propagation of events in γ_1, which is impossible because of the hypothesis 4, hence $\exists j \in \{i_1, \ldots, i_k\} | rcv(t_j) \in \Sigma_{rcv}^{\gamma_1}$.

Therefore, $(q_{i_1} \xrightarrow{t_{i_1}} q'_{i_1}, \ldots, q_{i_k} \xrightarrow{t_{i_k}} q'_{i_k})$ is a synchronised transition and belongs to $\|\gamma_1\|$. The fact that $(q_{j_1} \xrightarrow{t_{j_1}} q'_{j_1}, \ldots, q_{j_l} \xrightarrow{t_{j_l}} q'_{j_l})$ is a synchronised transition and belongs to $\|\gamma_2\|$, is shown in the same manner. The transition T is thus in E', hence the result. □

Acknowledgements

We thank the anonymous reviewers for their fruitful comments.

References

[1] A. Arnold. Transition systems and concurrent processes, in: G. Mirkowska, H. Rasiowa (Eds.), Mathematical problems in Computation theory, Banach Center, Warsaw, 1987, pp. 9–21.

[2] P. Baroni, G. Lamperti, P. Pogliano, M. Zanella, Diagnosis of large active systems, Artificial Intelligence, 110(1999) 135–183.

[3] P. Baroni and G. Lamperti and P. Pogliano and M. Zanella, Diagnosis of a class of distributed discrete-event systems, IEEE Transactions on Systems, Man, and Cybernetics, Part A, 30(6)(2000), pp. 731–752.

[4] R. E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation, IEEE Transactions on Computers, 35(8)(1986), pp. 677–691.

[5] E. M. Clarke, O. Grumberg, D. Peled, Model Checking. MIT Press, Cambridge, 1999.

[6] L. Console, C. Picardi, M. Ribaudo, Process algebra for systems diagnosis, Artificial Intelligence, 142(2002), pp. 19–51.

[7] M.-O. Cordier, C. Dousson, Alarm driven monitoring based on chronicles, in: Proceedings of Safeprocess'2000, Budapest, Hungary, 2000, pp. 286–291.

[8] R. Debouk, S. Lafortune, D. Teneketzis, Coordinated decentralized protocols for failure diagnosis of discrete event systems. Journal on Discrete-Event Dynamic Systems, 10(1-2)(2000), pp. 33–86.

[9] C. Dousson, P. Gaborit, M. Ghallab, Situation recognition: representation and algorithms, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-93), Chambéry, France, 1993, pp. 166–172.

[10] E. Fabre, A. Benveniste, C. Jard, Distributed diagnosis for large discrete event dynamic systems, in: Proceedings of the IFAC world congress, Barcelona, Spain, 2002.

[11] J. Kurien, P. P. Nayak, Back to the Future for Consistency-Based Trajectory Tracking, in: Proceedings of AAAI/IAAI, 2000, pp. 370–377.

[12] G. Lamperti and M. Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. Artificial Intelligence, 137(2002)(1–2), pp. 91–163.

[13] G. Lamperti and M. Zanella. Continuous diagnosis of discrete-event systems In *Proceedings of the International Workshop on Principles of Diagnosis (DX'03)*, pages 105–111, Washington, D.C., USA, 2003.

[14] G. Lamperti, M. Zanella, Diagnosis of active systems, Kluwer Academic Publishers, Dordrecht, 2003.

[15] J. Lunze, Discrete-event modelling and diagnosis of quantized dynamical systems, in: Proceedings of the International Workshop on Principles of Diagnosis (DX-99), Loch Awe, United Kingdom, 1999, pp. 147–154.

[16] A. Mazurkiewicz, Basic notions of trace theory, in: J. W. de Bakker, W.-P. de Roever, G. Rozenberg (Eds.), Proceedings of the School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Lecture Notes in Computer Science 354, Springer-Verlag, New York, 1988, pp. 364–397.

[17] D. Niebur, Expert systems for power system control in western Europe, in: Proceedings of the IEEE Symposium on Intelligent Control, Philadelphia, USA, 1990, pp. 112–119.

[18] D. Peled, All from one, one for all: on model checking using representatives, in: C. Courcoubetis (Ed.), Computer-Aided Verification, Lecture Notes in Computer Science 697, Springer-Verlag, New York, 1993, pp. 409–423.

[19] Y. Pencolé, Decentralized diagnoser approach: application to telecommunication networks, in: Proceedings of the International Workshop on Principles of Diagnosis (DX-00) , Morelia, Mexico, 2000, pp. 185–192.

[20] Y. Pencolé, M.-O. Cordier, L. Rozé, Incremental decentralized diagnosis approach for the supervision of a telecommunication network, in: Proceedings of the International Workshop on Principles of Diagnosis (DX-01), San Sicario, Italy, 2001, pp. 151–158.

[21] Y.Pencolé, M.-O. Cordier, L. Rozé, A decentralized model-based diagnostic tool for complex systems, International Journal on Artificial Intelligence Tools, 11(3)(2002) 327–346.

[22] M. Riese, Diagnosis of extended finite automata as a dynamic constraint satisfaction problem, in: Proceedings of the International Workshop on Principles of Diagnosis (DX-93), Aberystwyth, United Kingdom, 1993, pp. 60–73.

[23] L. Rozé, M-O. Cordier, Diagnosing discrete event systems: An experiment in telecommunication networks, in: Proceedings of the Workshop on Discrete Event Systems (WODES-98), Cagliari, Italy, 1998, pp. 130–137.

[24] L. Rozé, M.-O. Cordier, Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks, Journal on Discrete-Event Dynamic Systems, 12(1)(2002) 43–81, errata, 14(1)(2004) 131.

[25] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Diagnosability of discrete event system, IEEE Transactions on Automatic Control, 40(9)(1995) 1555–1575.

[26] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Active diagnosis of discrete-event systems, IEEE Transactions on Automatic Control 43(7)(1998) 908–929.

[27] R. Sengupta, Diagnosis and communication in distributed systems, in: Proceedings of the Workshop on Discrete Event Systems (WODES-98), Cagliari, Italy, 1998, pp. 144–151.

[28] B. C. Williams, P. P. Nayak, A model-based approach to reactive self-configuring systems, in: Proceedings of AAAI-96, 1996, pp. 971–978.

[29] B. C. Williams, P. P. Nayak, Immobile robots – AI in the new millenium, AI Magazine 17(3)(1996) 17–34.