Exploiting independence in a decentralised and incremental approach of diagnosis

Alban Grastien, Marie-Odile Cordier

Exploiting independence in a decentralised and incremental approach of diagnosis

Marie-Odile Cordier
Irisa Dream
Rennes – France
cordier@irisa.fr

Alban Grastien*
Irisa Dream
Rennes – France
alban.grastien@rsise.anu.edu.au

Abstract

It is now well-known that the size of the model is the bottleneck when using model-based approaches to diagnose complex systems. To answer this problem, decentralized/distributed approaches have been proposed. The global system model is described through its component models as a set of automata and the global diagnosis is computed from the component diagnoses (also called local diagnoses). Another problem, which is far less considered, is the size of the diagnosis itself. However, it can also be huge enough, especially when dealing with uncertain observations. It is why we recently proposed to slice the observation flow into temporal windows and to compute the diagnosis in an incremental way from these diagnosis slices.

In this context, we define in this paper two independence properties (transition and state independence) and we show their relevance to get a tractable representation of diagnosis. To illustrate the impact on the diagnosis size, experimental results on a toy example are given.

1 Introduction

In this paper, we are concerned with the diagnosis of discrete event systems [Cassandras and Lafortune, 1999] where the system behaviour is modeled by automata. This domain is an active domain since the seminal work proposed by [Sampath *et al.*, 1996]. It consists in finding what happened to the system from existing observations as in [Baroni *et al.*, 1999; Cordier and Thiébaux, 1994; Console *et al.*, 2000; Lunze, 1999; Rozé and Cordier, 1998; Cordier and Largouët, 2001]. A classical formal way of representing the diagnosis problem is to express it as the synchronised product of the system model automaton and an observation automaton. This formal definition hides the real problem which is to ensure an efficient computation of the diagnosis when both the system is complex and the observations possibly uncertain.

It is now well-known that the size of the system model is one bottleneck when using model-based approaches to diagnose complex systems. To answer this problem, decentralized/distributed approaches have been proposed [Pencolé and Cordier, 2005; Lamperti and Zanella, 2003; Benveniste *et al.*, 2005]. Instead of being explicitly given, the system model is described through its component models in a decentralized way. From these local models, local diagnoses are computed to explain local observations. When it is needed to take a global decision, a global diagnosis is computed by merging local diagnoses in order to take into account the synchronisation events which express the dependency relation which may exist between the components. This merging step can be costly and merging strategies have been proposed as in [Pencolé and Cordier, 2005; Lamperti and Zanella, 2003]. The main result gained from these work is the importance of detecting concurrent subsystems in order to limit both the computation time and the representation size of the diagnosis.

A problem, which is far less considered, is the size of the observation flow, which directly impact the size of the diagnosis itself. However, it can also be a problem, especially when dealing with uncertain observations as already remarked by [Lamperti and Zanella, 2003]. Moreover, increasing the observation period decreases the chance of finding independent subsystems. It is why we recently proposed to slice the observation flow into temporal windows and to compute the diagnosis in an incremental way from these diagnosis slices [Grastien *et al.*, 2005]. The idea is then to detect independent subsystems on these limited subperiods and to exploit these properties to get an economical representation and computation of diagnosis.

In this context of incremental and decentralised diagnosis, we define in Section 2 two independence properties (transition and state independence) on automata and we show their relevance to get a tractable representation of diagnosis. The first one, transition independence expresses that two models do not share any synchronisation events. The second one, state independence, expresses that when decomposing a model into two submodels, no constraints on their initial states have been lost. We first examine in Section 3 the purely decentralised case and propose to represent the diagnosis by a set of transition-independent diagnoses. We show in Section 4 the specific problem related to the incremental computation and propose to use an abstract description of trajectories, from which the set of final states and the trajectories of the global diagnosis can be easily retrieved. To illustrate the

*This work was partially made in NICTA, Canberra (Australia)

impact on our proposal on the diagnosis size, experimental results on a toy example are given in Section 5. We conclude in showing that the next step is to automatically find the best slicing points in order to maximally exploit the two independence properties which were defined.

2 Preliminaries and independence properties

We suppose in this paper that the behavioural models are described by automata. We thus begin by giving some definitions concerning automata which are needed in the following sections. Then, we define the independence properties that are central in this paper. Lastly, we recall the diagnosis definitions and state some hypotheses.

2.1 Automata, synchronisation and restriction

Automata are used to describe the behavioural models of the system components. Let us recall the definition and introduce the notations.

Definition 1 (Automaton)**.**
An automaton A *is a tuple* (Q, E, T, I, F) *where*

- Q *is a (finite) set of states,*
- E *is a set of labels,*
- $T \subseteq Q \times 2^E \times Q$ *is a (finite) set of transitions* (q, l, q')*, with* $l \subseteq E$*,*
- $I \subseteq Q$ *is the set of initial states,*
- $F \subseteq Q$ *is the set of final states.*

We suppose that $\forall q \in Q$, the transition $(q, \emptyset, q)$ exists. The label l on the transition $t = (q, l, q')$ indicates which events trigger the transition.

A trajectory is a path in the automaton joining an initial state to a final state.

Definition 2 (Trajectory)**.**
A trajectory *on an automaton* $A = (Q, E, T, I, F)$ *is a double sequence of states and transitions* $traj = q_0 \xrightarrow{l_1} \dots \xrightarrow{l_m} q_m$ *such that:* $q_0 \in I$*,* $q_m \in F$*, and* $\forall i,\ (q_{i-1}, l_i, q_i) \in T$*.*

The set of trajectories of an automaton A is denoted *Traj*(A). In the following, as we are interested mainly by trajectories and states passed through, the automata we consider are trim automata [Cassandras and Lafortune, 1999], i.e automata such that all the states belong at least to one trajectory. The trim operation transforms an automaton into its corresponding trim automaton by removing the states that do not belong to any trajectory. Remark that a trim operation does not remove any trajectory. It can however shrink the set of initial states and of final states.

Let us consider the trim automaton in Figure 1. The initial states are represented by an arrow with no origine state, and the final states by a double circle. Then, $1 \xrightarrow{\{a_1\}} 3 \xrightarrow{\{s_2\}} 5 \xrightarrow{\{a_1\}} 6$ is a trajectory.

Let us consider synchronisation of two automata A_1 and A_2. The events which are common the transition labels of A_1 and A_2, i.e. $E_1 \cap E_2$, are called synchronisation events. To be synchronizable, two transitions must either be labeled by events which are not synchronisation events, or have the same synchronisation events. The synchronisation operation on two automata builds the trim automaton where all the trajectories of both automata which cannot be synchronised according to the synchronisation events are removed.

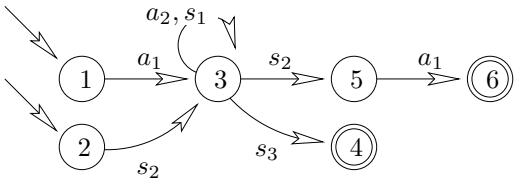

Figure 1: Example of automaton

Definition 3 (Synchronisation of automata)**.**
Given $A_1 = (Q_1, E_1, T_1, I_1, F_1)$ *and* $A_2 = (Q_2, E_2, T_2, I_2, F_2)$ *two automata. The* synchronised automaton *of* A_1 *and* A_2*, denoted* $A_1 \otimes A_2$*, is the trim automaton* $A = (Q_A, E, T_A, I_A, F_A) = Trim(Q, E, T, I, F)$ *such that:*

- $Q = Q_1 \times Q_2$*,*
- $E = E_1 \cup E_2$*,*
- $T = \{((q_1, q_2), l, (q_1', q_2')) \mid \exists l_1, l_2, (q_1, l_1, q_1') \in T_1 \wedge (q_2, l_2, q_2') \in T_2 \wedge (l_1 \cap (E_1 \cap E_2) = l_2 \cap (E_1 \cap E_2)) \wedge l = l_1 \cup l_2\}$*,*
- $I = I_1 \times I_2$*,*
- $F = F_1 \times F_2$*.*

The set of states Q_A is included in $Q_1 \times Q_2$ as some states (and transitions) can be removed by the trim operation. In the same way, the initial (resp. final) states of A, I_A (resp. F_A), are included in $I_1 \times I_2$ (resp. $F_1 \times F_2$).

Figure 2 gives an example of synchronisation. The automaton in Figure 1 and the automaton on the top of Figure 2 are synchronised leading to the automaton on the bottom. The synchronising events are the s_i events.

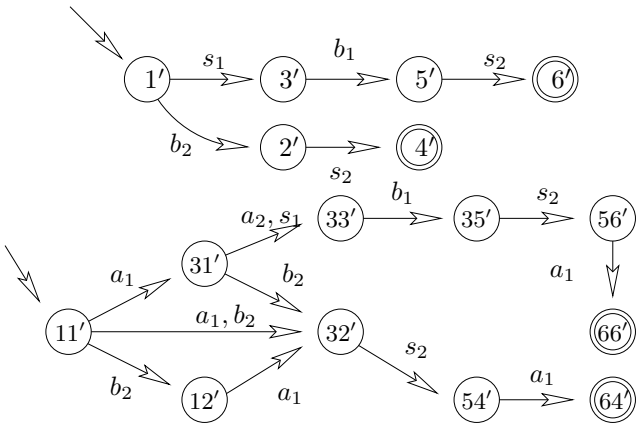

Figure 2: Example of synchronisation

The restriction operation of an automaton removes from I all the initial states which are not in the specified set of states.

Due to the trim operation, all the states and transitions which are no more accessible are removed from Q.

Definition 4 (Restriction)**.**
Let $A = (Q, E, T, I, F)$ be an automaton. The restriction *of A by the states of I', denoted $A[I']$, is the automaton $A' = (Q'_A, E, T'_A, I'_A, F'_A) = Trim(Q, E, T, I \cap I', F)$.*

2.2 Transition and State-independency

The transition-independency property states that two (or more) automata do not have any transition labeled with synchronisation events.

Definition 5 (Transition-independency)**.**
$A_1 = (Q_1, E_1, T_1, I_1, F_1)$ and $A_2 = (Q_2, E_2, T_2, I_2, F_2)$ are transition-independent (TI) *iff every label l on a transition of T_1 or T_2 is such that $l \cap (E_1 \cap E_2) = \emptyset$.*

For two TI automata, the synchronisation operation is equivalent to a shuffle operation.

Property 1: Let A_1 and A_2 be two transition-independent automata and $A = A_1 \otimes A_2$. The final (resp. initial) states of A correspond exactly to the Cartesian product of the final (resp. initial) states of A_1 and A_2.

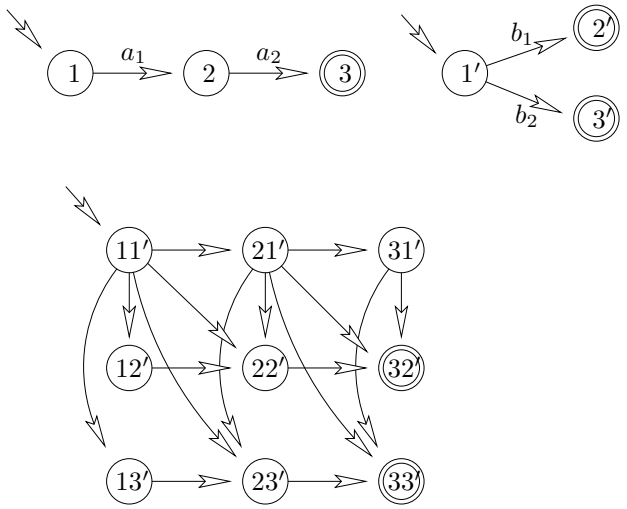

Figure 3: Example of two TI automata

Figure 3 gives an example of two automata A_1 and A_2 that synchronise on s_i events. Since none of the automata has a transition labeled with a s_i event, the automata are transition-independent. The synchronisation $A = A_1 \otimes A_2$ is represented on the bottom of the figure (for simplicity, the labels on the transitions are not represented). We see that the set of initial states I (resp. F) is the Cartesian product of I_1 (F_1) and I_2 (F_2). Figure 1 and Figure 2 give an exemple of two automata A_1 and A_2 that are not transition-independent as they contain transitions with s_i events. The set of final states F of the synchronisation is only included in the Cartesian product of F_1 and F_2.

In the next section 3, we are interested in representing a system model in a decomposed way by the set of its subsystems models, the main property being that it must be possible to retrieve the first one from the other ones by a composition (synchronisation) operation. In the following, we give the definitions of subsystems and the properties the set of subsystems models must satisfy to get a safe representation of the system model.

Definition 6 (System and subsystems)**.**
A system *can be described by its set of components Γ. A* subsystem *s is a non-empty set of components: $s \subseteq \Gamma$.*

A subsystem model describes the subsystem behaviour and is described by an automaton $A_s = (Q_s, E_s, T_s, I_s, F_s)$ where E_s is the set of events that can occur on this subsystem. Some of these events are shared with other subsystems and are synchronisation events between subsystem models.

Let us now see the properties a set of subsystem models has to satisfy to be a good representation of the system model. We first define what we call a decomposition of A in two automata.

Definition 7 (Decomposition of A)**.**
Two automata A_1 and A_2 are said to be a decomposition *of an automaton A iff $A = (A_1 \otimes A_2)[I]$ where I are the initial states of A.*

Remark that we do not require that we get exactly A by synchronizing A_1 and A_2, but only a super-automaton of A (i.e. an automaton that contains all the trajectories of A and possibly more). In general, we have thus that the initial (resp. final) states of A are included in $I_1 \times I_2$ (resp. $F_1 \times F_2$).

The idea is that, when you describe a system (whose model is A) by its subsystems, you have to describe the subsystem behaviours, which is done through the subsystem models (here A_1 and A_2) and the way the subsystems interact, which is done through the synchronisation events. Moreover, you have to do it in a proper way given by the Definition 7. But a point is still missing, as the constraints existing between the subsystem initial states in order to represent the system initial states can be lost in the decomposition of A. It is why, when composing A_1 and A_2 by a synchronisation operation, we do not get always back exactly A, but an automaton including A.

The state-independency property is a property of a decomposition A_1 and A_2 which ensures that we get exactly A.

Definition 8 (State-independency decomposition wrt A)**.**
Two automata $A_1 = (Q_1, E_1, T_1, I_1, F_1)$ and $A_2 = (Q_2, E_2, T_2, I_2, F_2)$ are said to be a state-independent decomposition wrt A (SI_A) *iff they are a decomposition of A and if $A = A_1 \otimes A_2$.*

Remark that, if A_1 and A_2 have both a unique initial state, and if they are a decomposition wrt A, then A_1 and A_2 are a state-independent decomposition wrt A.

Let us suppose two automata A_1 and A_2 which are a state-independent decomposition wrt A and are both transition-independent. In this case, due to Property 1, the initial and final states of A can be easily computed as the Cartesian product of the initial and final states of A_1 and A_2. This property means that, when you are mainly interested in these states, you do not have to perform the synchronisation operation on the automata, which is costly in space.

Property 2: Let A_1 and A_2 be two transition-independent automata forming a state-independent decomposition wrt A. The initial (resp. final) states of A are exactly the Cartesian product of the initial (resp. final) states of A_1 and A_2.

When A_1 and A_2 are not a state-independent decomposition wrt A, the only way not to lose any information is to add as extra information the initial states of A to the decomposed representation of A.

2.3 Diagnosis

Let us recall now the definitions used in the domain of discrete-event systems diagnosis where the model of the system is represented by an automaton. t_0 corresponds to the starting time and t_n to the ending time of diagnosis.

Definition 9 (Model)**.**
The model of the system*, denoted Mod, is an automaton.*

The model of the system describes its behaviour and the trajectories of *Mod* represent the evolutions of the system. The set of initial states I^{Mod} is the set of possible states at t_0. We suppose as usual that $F^{Mod} = Q^{Mod}$ (all the states of the system may be final). The set of observable events is denoted $E^{Mod}_{Obs} \subseteq E^{Mod}$.

Let us turn to observations represented by an automaton, where the transition labels are observable events of E^{Mod}_{Obs}.

Definition 10 (Observation automaton)**.**
The observation automaton*, denoted Obs, is an automaton describing the observations emitted by the system during the period* $[t_0, t_n]$.

Even if usually the observations are subject to uncertainties, we consider in the following that they are represented as a unique sequence of observable events. It allows us to simplify the presentation but it can be extended to the case of uncertain observations as we did for instance in [Grastien *et al.*, 2005].

The *diagnosis*, denoted Δ, is a trim automaton describing the possible trajectories on the model of the system compatible with the observations sent by the system during the period $[t_0, t_n]$. The diagnosis is then formally defined as resulting from the synchronisation operation between the system model *Mod* and the observation automaton *Obs* .

Definition 11 (Diagnosis)**.** *The* diagnosis*, denoted* Δ*, is a trim automaton such that* $\Delta = Mod \otimes Obs$

3 Improving diagnosis representation in a decentralised approach

Real-world systems can often be seen as a set of (possibly abstract) interconnected components. Each component has a simple behaviour but the connections between the components can lead to a complex global behaviour. For this reason, the size of a global model of the system is generally untractable and no global model can be effectively built. To answer this problem, decentralised/distributed approaches have been proposed [Lamperti and Zanella, 2003; Pencolé and Cordier, 2005; Benveniste *et al.*, 2005]. In this article, we consider the decentralised approach of Pencolé and Cordier. This approach is pictured on Figure 4.

The idea is to describe the system behaviour in a decomposed way. The so-called *decentralised* model is thus d*–Mod* $= \{Mod_1, \ldots, Mod_n\}$ where Mod_i is the behavioural model of the component $\mathcal{C}_i$. The decentralised model is built to be a decomposition of the global model *Mod*. The global model can thus be retrieved by $Mod = (Mod_1 \otimes \ldots \otimes Mod_n)[I]$ where I is the set of initial states of *Mod*. We consider that the global model has a unique initial state (if it is not, an initialization transition can be added to ensure it) and that the component models have also a unique initial state. They are thus a state-independent decomposition wrt *Mod* and we have $Mod = Mod_1 \otimes \ldots \otimes Mod_n$.

The observations *Obs* can generally be decentralised as follows: d*–Obs* $= \{Obs_1, \ldots, Obs_n\}$ such that Obs_i contains the observations from the component $\mathcal{C}_i$ and such that: $Obs = Obs_1 \otimes \ldots \otimes Obs_n$.

Given the local model Mod_i and the local observations Obs_i, it is possible to compute the local diagnosis $\Delta_{\mathcal{C}_i} = Mod_i \otimes Obs_i$. These diagnoses represent the local behaviours that are consistent with the local observations. It was shown in [Pencolé and Cordier, 2005] that the decentralised diagnosis is a decomposition of Δ. As there is a unique initial state, it is also a state-independent decomposition. It is then possible to compute the global diagnosis of the system by *merging* all the local diagnoses as follows: $\Delta = \Delta_{\mathcal{C}_i} \otimes \ldots \otimes \Delta_{\mathcal{C}_n}$.

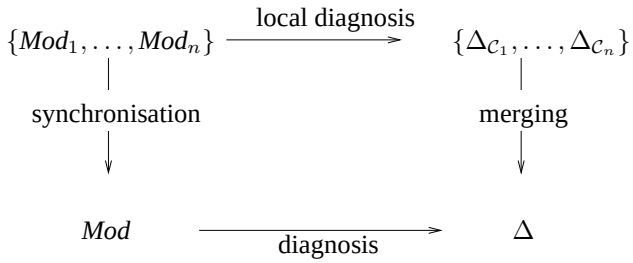

Figure 4: Principle of the decentralised computation of the diagnosis

A first improvement in the diagnosis computation is that, rather than directly merging all the local diagnoses together, it is possible to incrementally compute the global diagnosis by successive synchronisation operations. Let s_1 and s_2 be two disjoint subsystems (possibly being components) and let $s = s_1 \uplus s_2$ be the subsystem that contains exactly s_1 and s_2. The subsystem diagnosis Δ_s can be computed by synchronising the two subsystem diagnoses Δ_{s_1} and Δ_{s_2}: $\Delta_s = \Delta_{s_1} \otimes \Delta_{s_2}$. The diagnosis of the system Γ is $\Delta = \Delta_\Gamma$.

The next point is that, in spite of the constraints generated by the observations, the size of the global diagnosis can still be large. It is mainly due to the fact that merging concurrent diagnoses corresponds to compute the shuffle of two automata which is costly in terms of number of states and transitions (see for instance Figure 3). A second improvement to

avoid these costful shuffles is to represent the system diagnosis as a set of transition-independent subsystem diagnoses.

Definition 12 (Decentralised diagnosis)**.**
A decentralised diagnosis d–Δ *is a set of subsystem diagnoses* $\{\Delta_{s_1}, \ldots, \Delta_{s_k}\}$ *such that*

- Δ_{s_i} *is the diagnosis of the subsystem* s_i*,*
- $\{s_1, \ldots, s_k\}$ *is a partition of the system* Γ*, and*
- $\forall i, j \in \{1, \ldots, k\}$, $i \neq j \Rightarrow \Delta_{s_i}$ *and* Δ_{s_j} *are transition-independent.*

As seen before, a decentralised diagnosis d–$\Delta = \{\Delta_{s_1}, \ldots, \Delta_{s_k}\}$ is a decomposition of the global diagnosis. It can thus be computed, if needed, by synchronising all the subsystem diagnoses, or equivalently by a shuffle operation as $\Delta = \Delta_{s_1} \otimes \ldots \otimes \Delta_{s_k}$. Its final states can be obtained by a simple Cartesian product on the final states of all Δ_{s_i}.

Algorithm 1 shows how to compute the decentralised diagnosis from the local (component) diagnoses. Until all pairs of diagnoses are transition-independent, the algorithm chooses two transition-dependant diagnoses and merges them. Let us remark that the result is not unique and depends on the merging strategy which is also very important from a computation time point of view. It was proposed in [Pencolé *et al.*, 2001a] to use a dynamic strategy, based on first synchronising the subsystem diagnoses which interact the most, in order to remove at first as many trajectories as possible.

Algorithm 1 Algorithm to compute a decentralised diagnosis

input: local diagnoses $\{\Delta_{\mathcal{C}_1}, \ldots, \Delta_{\mathcal{C}_n}\}$
d–$\Delta = \{\Delta_{\mathcal{C}_1}, \ldots, \Delta_{\mathcal{C}_n}\}$
while $\exists \Delta_{s_1}, \Delta_{s_2} \in d$–$\Delta$ such that Δ_{s_1} and Δ_{s_2} are not transition-independent **do**
  d–$\Delta = d$–$\Delta \setminus \{\Delta_{s_1}, \Delta_{s_2}\}$
  $s = s_1 \uplus s_2$
  $\Delta_s = \Delta_{s_1} \otimes \Delta_{s_2}$
  d–$\Delta = d$–$\Delta \cup \{\Delta_s\}$
end while
return: d–Δ

4 Improving diagnosis representation in a decentralised and incremental approach

In the previous section, we considered that the diagnosis was computed on a period. This means that the observation automaton represents the observations from the beginning to the end of the period, and the diagnosis represents the behaviour during the whole period.

We have seen in the previous section that exploiting transition-independence enables to reduce the size of the diagnosis representation. However, when we consider a long period, as this may be the case when you have log files to diagnose, it is very seldom that you have independent behaviours since each component eventually interacts with most of its neighbours. It is why we recently proposed to *slice* the observations into temporal windows and to incrementally compute the diagnosis for each temporal window [Grastien *et al.*, 2005]. Given these diagnoses on *small* windows, it can now be expected to have independent behaviours that can be efficiently represented by a decentralised diagnosis.

The problem with the incremental approach is that it becomes difficult to ensure the state-independency property of the decomposition. This property allowed us, due to Property 2 of Section 2, to get the initial and final states of the global diagnosis without computing it explicitly. To keep the benefit of the decentralised representation of diagnosis, we propose a solution that enables us to get the initial and final states needed for an incremental diagnosis without having to merge diagnoses, even when state-independency is not satisfied.

Let us first present a formalism-free generalization of the incremental computation by automaton slicing. We explain then why we lose the state-independency property and end by proposing a solution to this problem.

4.1 Incremental diagnosis

The incremental diagnosis relies on the notion of temporal windows first introduced in [Pencolé *et al.*, 2001b]. For a detailed presentation of the diagnosis by slices, refer to [Grastien *et al.*, 2005]. Let $[t_0, t_n]$ be the diagnosis period and $t_0, \ldots, t_n$ be a sequence of dates. The temporal window $\mathcal{W}^i$ is the period $[t_{i-1}, t_i]$. Let $\mathit{Obs}^1, \ldots, \mathit{Obs}^n$ be a slicing of the observations *Obs*. It is shown in [Grastien *et al.*, 2005] that, given a slicing of the observations $\mathit{Obs}_n = \mathit{Obs}^1, \ldots, \mathit{Obs}^n$, a diagnosis Δ_n on the period $[t_0, t_n]$ can be computed as a sequence of n diagnoses $(\Delta^1, \ldots, \Delta^n)$ corresponding to the n windows $\mathcal{W}^i$. It is also shown that, given this sequence of automata, it is possible, only if needed, to *reconstruct* the original automaton Δ_n by appending the slices. The trajectories can be computed as follows: A trajectory on this sequence of automata is a sequence of n trajectories $\mathit{traj}^i = q_0^i \xrightarrow{l_1^i} \ldots \xrightarrow{l_{mi}^i} q_{mi}^i \in \mathit{Traj}(A^i)$ where $\forall i$, $q_0^{i+1} = q_{mi}^i$.

Let us reduce now the problem to two slices and suppose we have computed a diagnosis Δ_{i-1} for the period $[t_0, t_{i-1}]$. We do not presume the way this diagnosis is represented and will come back on this point later. We want to compute the diagnosis Δ_i by taking into account the observations Obs^i on the next temporal window $\mathcal{W}^i$. Let us first see how the diagnosis Δ^i can be computed. We can state that

- $\Delta^1 = \mathit{Mod} \otimes \mathit{Obs}^1$, and
- $\forall i \neq 1$, $\Delta^i = (\mathit{Mod}^- \otimes \mathit{Obs}^i)[F_\Delta^{i-1}]$ where $\mathit{Mod}^- = (Q^{\mathit{Mod}}, E^{\mathit{Mod}}, T^{\mathit{Mod}}, Q^{\mathit{Mod}}, F^{\mathit{Mod}})$ and F_Δ^{i-1} is the set of final states of Δ^{i-1}.

The ith diagnosis of the sequence can be theoretically computed by the synchronisation of the model (where all states are initial Mod^-) with the observations Obs^i of the window. It is however important from a computational point of view to restrict the set of initial states with the set of final states F_Δ^{i-1} of the previous automaton. It is then possible to describe Δ_i as the sequence Δ_{i-1}, Δ^i. Remark that the set of final states of Δ_i is exactly the set of final states of Δ^i.

4.2 Loss of the state-independency property

Our goal is to use, for this sequence of diagnoses, a decentralised computation based on a decentralised model, and a decentralised representation similar to the one proposed in Section 3 based on transition-independent diagnoses.

We want to compute Δ^i in a decentralised way, which means that we build the local diagnoses before merging them (see Algorithm 1). The diagnosis of the component $\mathcal{C}$ in the temporal window $\mathcal{W}^i$ is computed as follows : $\Delta^i_{\mathcal{C}} = (\mathit{Mod}^-_{\mathcal{C}} \otimes \mathit{Obs}^i)[F^{i-1}_{\Delta} \downarrow \mathcal{C}]$ where $[F^{i-1}_{\Delta} \downarrow \mathcal{C}]$ is the projection operation of the final states of Δ_{i-1} on component $\mathcal{C}$.

By Algorithm 1, we get a set of transition-independent subsystem diagnoses. The problem that appears here is that this set is a decomposition of Δ_i, but it can not be ensured that it is a state-independent decomposition. Contrary to the case of Section 3, it can be the case that existing links with the initial states of the other components are lost when projecting F^{i-1}_{Δ} on a component $\mathcal{C}$.

This is illustrated by Figure 5. The figure represents the diagnosis of two components. These components can be either in a Ok state or Faulty state. The figure presents a two-window diagnosis, each in a box. During the first window, one of the two components failed but it is not possible to determine which component did. The initial states of each component at the beginning of the second window are obtained by projecting the final states of the first window and they are O and F for one component and O' and F' for the other one. Nothing happened during the second window. The algorithm proposes thus the two local diagnoses (up and bottom in Figure 5, right) but we can see that the links between the initial states were lost during the projection, and then we get a decomposition of the global diagnosis which is not state-independent. We have $F^2_{\Delta} \subset \{O, F\} \times \{O', F'\}$. To get the exact final states F^2_{Δ}, the only solution would be to synchronize the local diagnoses and then to use the restriction operation with the final states of the first window as argument, which is not an economical way as expected. We propose below a solution to this problem.

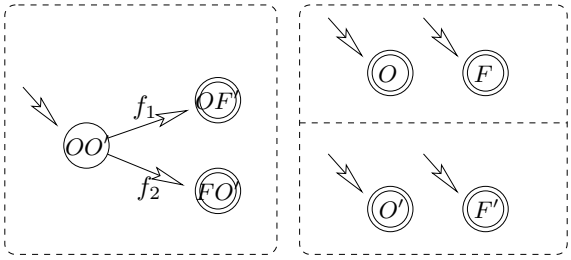

Figure 5: Example of loss of information in a naive decentralised representation of the incremental diagnosis

4.3 TI + abstract representation

The solution we propose is to add an abstract representation of the diagnosis to the set of transition-independent subsystem diagnoses. We first define what is an abstration, and then show that it allows us to keep the benefit of the decentralised representation even when it is not state-independent wrt the global diagnosis as shown in 4.2.

An abstraction of an automaton only preserves as states the initial and final states of the original automaton, and abstracts the trajectories existing in the original automaton in a transition labeled by $\emptyset$.

Definition 13 (Abstraction)**.**
Let $A = (Q, E, T, I, F)$ be a trim automaton. The abstraction *of A, denoted $\mathit{Abst}(A)$, is the (trim) automaton $A' = (Q', E', T', I', F')$ where:*

- $Q' = I \cup F$,
- $E' = \emptyset$,
- $T = \{(q, \emptyset, q') \mid \exists \mathit{traj} = q_0 \xrightarrow{l_1} \ldots \xrightarrow{l_n} q_n \in \mathit{Traj}(A) \wedge q_0 = q \wedge q_n = q'\}$,
- $I' = I$, *and*
- $F' = F$.

The following two properties can be easily proved.
Property 3: Let A_1 and A_2 be two transition-independent automata. Then, $\mathit{Abst}(A_1) \otimes \mathit{Abst}(A_2) = \mathit{Abst}(A_1 \otimes A_2)$.
Property 4: Let A_1 and A_2 be two transition-independent automata and let I be a set of states. Then, $(\mathit{Abst}(A_1) \otimes \mathit{Abst}(A_2))[I] = \mathit{Abst}((A_1 \otimes A_2)[I])$.

The main problem with the loss of the state-independency property is that we can no longer get the set of final states by a mere Cartesian product on the final states of the subsystem diagnoses. The abstraction allows us to compute them without having to perform the expensive synchronisation of the subsystems diagnoses. In fact, the final states are directly computed as the Cartesian product of the final states of the abstraction of the subsystems diagnoses which is a lot less expensive.

Let us consider the ith window $\mathcal{W}^i$. We know the set I^i of initial states of the current window as they are the final states of the preceding one. This set can be in a decentralised form, *ie* described by a set of states $\{I^i_1, \ldots, I^i_p\}$ such that $I^i = I^i_1 \times \ldots \times I^i_p$. As explained in 4.2, the subsystem diagnoses are computed using Algorithm 1 which returns a set d–$\Delta^i = \{\Delta^i_{s_1}, \ldots, \Delta^i_{s_k}\}$ of transition-independent diagnoses. We need to get the final states as they are used to restrict the initial states of the next window, but in absence of state-independency property, they can no longer be computed from the final states of d–Δ^i (in fact $F^i_{\Delta} \subseteq F_{d\text{–}\Delta^i}$).

To build the abstract representation, we propose to use Algorithm 2. To obtain the set of final states, the idea is, instead of synchronising the transition-independent automata, to synchronise their abstractions. Then, a restriction is performed using the initial states I^i, to get the exact final states F^i.

As at the end all the abstract subsystem diagnoses composing $Abstd$–Δ^i are state-independent, we know that the set of initial states of Δ^i is the set of initial states of $Abstd$–Δ^i. Moreover, we have the following property :
Property 5: The set of final states of Δ_i is the set of final states of $Abstd$–Δ^i.

Algorithm 2 Computation of the abstract representation of the diagnosis of Δ^i

input: local diagnoses $d\text{–}\Delta^i = \{\Delta^i_{s_1}, \ldots, \Delta^i_{s_k}\}$ + the set I of initial states
$Abstd\text{–}\Delta^i = \{Abst(\Delta^i_{s_j}) \mid \forall \Delta^i_{s_j} \in d\text{–}\Delta^i\}$
while $\exists Abst\Delta^i_{s_1}, Abst\Delta^i_{s_2} \in Abstd\text{–}\Delta^i$ such that $Abst\Delta^i_{s_1}$ and $Abst\Delta^i_{s_2}$ are not state-independent wrt $(Abst\Delta^i_{s_1} \otimes Abst\Delta^i_{s_2})[I \downarrow s_1 \uplus s_2]$ **do**
  $Abstd\text{–}\Delta^i = Abstd\text{–}\Delta^i \setminus \{Abst\Delta^i_{s_1}, Abst\Delta^i_{s_2}\}$
  $s = s_1 \uplus s_2$
  $Abst\Delta^i_s = (Abst\Delta^i_{s_1} \otimes Abst\Delta^i_{s_2})[I \downarrow s_1 \uplus s_2]$
  $Abstd\text{–}\Delta^i = Abstd\text{–}\Delta^i \cup \{Abst\Delta^i_s\}$
end while
return: $Abstd\text{–}\Delta^i$

It is then possible to get the set of final states of Δ_i without synchronising the transition-independent subsystem diagnoses. The decentralised representation of diagnosis on a temporal window is thus the set of its transition-independent subsystem diagnoses and the set of its transition and state-independent abstract diagnoses.

5 Experiments

In this section, we present an experimentation of the diagnosis using the decentralised and incremental approach. We present the system to diagnose and then give the results.

5.1 System

The system we want to diagnose is a network of 14 interconnected components as presented on Figure 6.

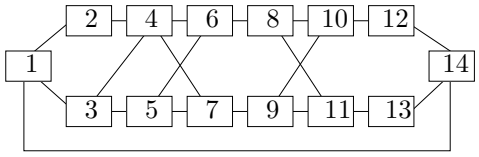

Figure 6: Topology of the network

Each component has the same behaviour: when a fault occurs on a component, it reboots and forces its neighbours to reboot too. When asked to reboot, the component sends the observation $IReboot_i$ (where i is the number of the component), and when the reboot is finished, it sends the observation $IAmBack_i$. When a component is asked to reboot, it can be asked to reboot by another component (and then send the $IReboot_i$ observation) at the beginning of the rebooting process.

The model is presented Figure 7. The *reboot!* message indicates that *reboot* is sent to all the neighbours, and the *reboot?* message indicates that a neighbour sent the *reboot* message to the component. So, for example, on component 1, there are three transitions from state O to state R respectively labeled by $\{reboot_{14\to1}, IReboot_1\}$, $\{reboot_{2\to1}, IReboot_1\}$, and $\{reboot_{3\to1}, IReboot_1\}$ since components 2, 3 and 14 are neighbours from component 1.

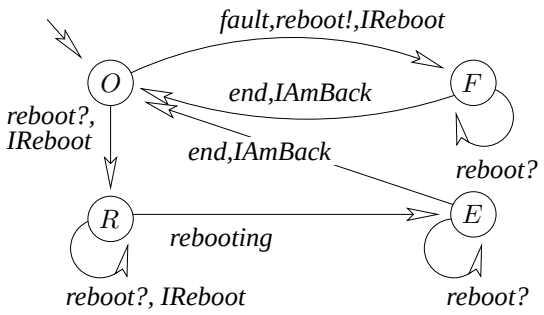

Figure 7: Model of a component

Let us remark that the decentralised modeling contains exactly 4×14 states, while the global model would contain nearly $4^{14} \approx 250\,000\,000$ states.

5.2 Results

The algorithms were programmed in Java, and run on a Linux machine with a 1.73 GHz Intel processor. We deal with 45 observations. The experiments results are given Table 1.

The first experiment was made with a unique temporal window as presented in section 3. The computation was more than 26 mn and produced 4 automata, one of which contains 9 085 states and 557 836 transitions. It can be noted that taking into account the transition-independence property of diagnoses in the decentralised representation is interesting as four independent subsystems are identified. It prevents from computing the shuffle for these subsystem diagnoses which is certainly a very good point. However, due to the length of the window, one of the automata is still very large.

Using the method described in section 4, the observations are now sliced into 4 temporal windows. The diagnosis was computed in less that 1 second, producing 39 small automata. The number of states is 479, that is 5% of the number of states used in the previous automaton, and the number of transitions is 4 038 which represents less than 1% of the transitions of the previous automaton. It confirms that slicing observations is beneficial in that it allows to increase the number of independent subsystems, and thus diagnoses.

	no slicing	1st slicing	2nd s.	3rd s.
nb states	9 088	479	589	3 375
nb trans	557 836	4 038	5 382	142 517
nb auto	4	39	51	26
time	26mn 55s	$<$ 1s	10 s	3mn 5s

Table 1: Results of the experimentations

Let us stress now the importance of the slicing on the good results of the method. In a third experiment, the first temporal window of the previous experiment was sliced into two. It can be noted that the number of states of the diagnosis increased by about 20% and the number of transitions by 30%. Moreover, the computation time increased to 10 seconds. The reason is that you sometimes need to have enough observations on a subsystem to conclude that this subsystem did not communicate with another subsystem.

In a fourth experiment, two temporal windows of the first window are merged into one unique window. The corresponding computation time is then nearly 4 minutes and the number of states and transitions exploded. It confirms that the slicing operation is a critical operation and that deciding what is the best slicing is an appealing perspective.

6 Conclusion

In this paper, we consider the diagnosis of discrete-event systems modeled by automata. To avoid the state-explosion problem that appears when dealing with large systems, we use a decentralised computation of the diagnosis. This approach consists in dividing the system into transition-independent subsystems. We show that the global diagnosis can be safely represented by the set of diagnoses of these transition-independent subsystems. An important point is that the transitions can be easily computed from this decentralised representation by relying on the *state-independency* property which we define. It is then clear that the smaller the transition-independent subsystems are, the best the diagnosis computation is, both according to time and space efficiency.

When the period of observation is important, very seldom do you have independent subsystems, since each component eventually interacts with most of its neighbours. We propose thus to slice the diagnosis period into temporal windows, in order to get, on these windows, transition-independent subsystems. The problem that appears is that the state-independency property does not hold anymore. We are then no more able to get the exact final states. On the one hand, such a set of diagnoses for transition-independent but not state-independent subsystems gives us only a superset of the global diagnosis, which is not satisfying. On the other hand, computing the set of transition-independent and state-independent subsystem diagnoses would be too expensive. We thus propose to keep the decentralised diagnosis representation (a set of transition-independent subsystem diagnoses), and to add an abstract representation of both state- and transition-independent diagnoses, enabling us to compute in an economic and efficient way the final states. We show that we get a safe representation of the global diagnosis.

Some points need to be analysed in more details. As can be seen in Algorithm 2, it is necessary to have an efficient way to check whether two abstract diagnoses are or not state-independent, and we are currently working on this point. Another concern is about the slicing. As shown in section 5, a *bad* slicing can lead to a very little benefit. An interesting prospect would be to automatically find the best slicing to obtain a diagnosis represented as efficiently as possible.

In this article, we considered that the observations were sure and ordered. In real-world systems, this hypothesis generally does not hold, and we proposed to represent the observation by an automaton [Grastien *et al.*, 2005]. The results of this article can be extended to cope with that. A more difficult case to consider is when you have to slice *on-line* the observations, while not all the observations are yet received. Finally, since we deal with state-spaces that are different from a window to the next, it should be interesting to use these results for reconfigurable systems, the topology (the set of components and the connections between them) of which can evolve along time, as considered for instance in [Grastien *et al.*, 2004].

References

[Baroni *et al.*, 1999] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence*, 110:135–183, 1999.

[Benveniste *et al.*, 2005] A. Benveniste, S. Haar, E. Fabre, and Cl. Jard. Distributed monitoring of concurrent and asynchronous systems. *Discrete Event Dynamic Systems*, 15(1):33–84, 2005.

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

[Console *et al.*, 2000] L. Console, C. Picardi, and M. Ribaudo. Diagnosis and diagnosability using PEPA. In *ECAI'2000*, pages 131–135, 2000.

[Cordier and Largouët, 2001] M.-O. Cordier and Ch. Largouët. Using model-checking techniques for diagnosing discrete-event systems. In *Twelfth International Workshop on Principles of Diagnosis (DX-01)*, pages 39–46, 2001.

[Cordier and Thiébaux, 1994] M.-O. Cordier and S. Thiébaux. Event-based diagnosis for evolutive systems. In *DX'1994*, pages 64–69, 1994.

[Grastien *et al.*, 2004] A. Grastien, M.-O. Cordier, and Ch. Largouët. Extending decentralized discrete-event modelling to diagnose reconfigurable systems. In *Fifteenth International Workshop on Principles of Diagnosis (DX-04)*, pages 75–80, 2004.

[Grastien *et al.*, 2005] A. Grastien, M.-O. Cordier, and Ch. Largouët. Incremental diagnosis of discrete-event systems. In *Sixteenth International Workshop on Principles of Diagnosis (DX-05)*, pages 119–124, 2005.

[Lamperti and Zanella, 2003] G. Lamperti and M. Zanella. *Diagnosis of Active Systems*. Kluwer Academic Publishers, 2003.

[Lunze, 1999] J. Lunze. Discrete-event modeling and diagnosis of quantized dynamical systems. In *10th International Workshop on Principles of Diagnosis (DX-99)*, pages 147–154, 1999.

[Pencolé and Cordier, 2005] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence Journal*, 164(1-2):121–170, 2005.

[Pencolé *et al.*, 2001a] Y. Pencolé, M.-O. Cordier, and L. Rozé. A decentralized model-based diagnostic tool for complex systems. In *The Thirteenth IEEE international conference on tools with artificial intelligence (ICTAI'01)*, pages 95–102, 2001.

[Pencolé *et al.*, 2001b] Y. Pencolé, M.-O. Cordier, and L. Rozé. Incremental decentralized diagnosis approach for the supervision of a telecommunication network. In *Twelfth International Workshop on Principles of Diagnosis (DX-01)*, pages 151–158, 2001.

[Rozé and Cordier, 1998] L. Rozé and M.-O. Cordier. Diagnosing discrete-event systems: an experiment in telecommunication networks. In *Fourth International Workshop on Discrete Event Systems (WODES'98)*, 1998.

[Sampath *et al.*, 1996] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *Control Systems Technology*, 4(2):105–124, 1996.