Exploiting Independence in a Decentralised and Incremental Approach of Diagnosis

Marie-Odile Cordier
University of Rennes1 / Irisa
Rennes – France
cordier@irisa.fr

Alban Grastien[*]
National ICT Australia
Canberra – Australia
alban.grastien@nicta.com.au

Abstract

It is well-known that the size of the model is a bottleneck when using model-based approaches to diagnose complex systems. To answer this problem, decentralised/distributed approaches have been proposed. Another problem, which is far less considered, is the size of the diagnosis itself. However, it can be huge enough, especially in the case of on-line monitoring and when dealing with uncertain observations. We define two independence properties (state and transition-independence) and show their relevance to get a tractable representation of diagnosis in the context of both decentralised and incremental approaches. To illustrate the impact of these properties on the diagnosis size, experimental results on a toy example are given.

1 Introduction

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 very active since the seminal work by [Sampath *et al.*, 1996]. Its aim is to find what happened to the system from existing observations. 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 a bottleneck when using model-based approaches to diagnose complex systems. To answer this problem, decentralised/distributed approaches have been proposed [Pencolé and Cordier, 2005; Lamperti and Zanella, 2003; Benveniste *et al.*, 2005; Jiroveanu and Boël, 2005]. Instead of being explicitly given, the system model is described through its component models in a decentralised way. The 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.

A problem, which is far less considered, is the size of the diagnosis itself. However, it can also be a problem, especially when dealing with a long-lasting flow of uncertain observations as remarked in [Lamperti and Zanella, 2003]. We define in this paper two independence properties (*state-independence* and *transition-independence*) and show that they make it possible to get a decentralised representation of the diagnosis without any loss of information. When the diagnosis is performed on a long-lasting temporal window, the transition independence property, which expresses the concurrence between subsystems, has little chance of being satisfied. The idea then is to *slice* the diagnosis period into smaller periods and to compute the diagnosis in an incremental way. The problem which appears is that the slicing generally removes the state-independence property. The solution we finally propose is to enrich the representation with an abstract description of the diagnosis trajectories and show that it allows us to keep the benefits of the transition-independence based decentralised representation.

After preliminaries in Section 2, two independence properties (state and transition-independence) on automata are defined in Section 3. In Section 4, the decentralised approach is proposed and the decisive role played by the *state-independence* property in the diagnosis computation is highlighted. Then it is shown how to take advantage of the *transition-independence* property to get an efficient representation of the diagnoses. The incremental computation and the use of an abstract representation is considered in Section 5. To illustrate the impact of our proposal on the diagnosis size, experimental results on a toy example are given in Section 6.

2 Preliminaries

We suppose in this paper that the behavioural models are described by automata. We thus begin by giving some definitions concerning automata (see [Cassandras and Lafortune, 1999]). Then, we recall the diagnosis definitions and state some hypotheses.

2.1 Automata, synchronisation and restriction

An *automaton* is a tuple (Q, E, T, I, F) where Q is the set of states, E the set of events, T the set of transitions (q, l, q') with $l \subseteq E$, and I and F the sets of initial and final states. For each state $q \in Q$, $(q, \emptyset, q) \in T$.

[*] This work was mainly made when the author was Ph.D student at University of Rennes1/Irisa, Rennes, France.

A *trajectory* is a path in the automaton joining an initial state to a final state. The set of trajectories of an automaton A is denoted *Traj*(A). In the following, we consider *trim* automata. The trim operation transforms an automaton by removing the states that do not belong to any trajectory.

Example: Let us consider the first trim automaton in Figure 1. The initial states are represented by an arrow with no origin state, and the final states by a double circle. Then, $1 \xrightarrow{\{a_1\}} 3 \xrightarrow{\{a_2,c_1\}} 3 \xrightarrow{\{c_3\}} 4 \xrightarrow{\emptyset} 4$ is a trajectory.

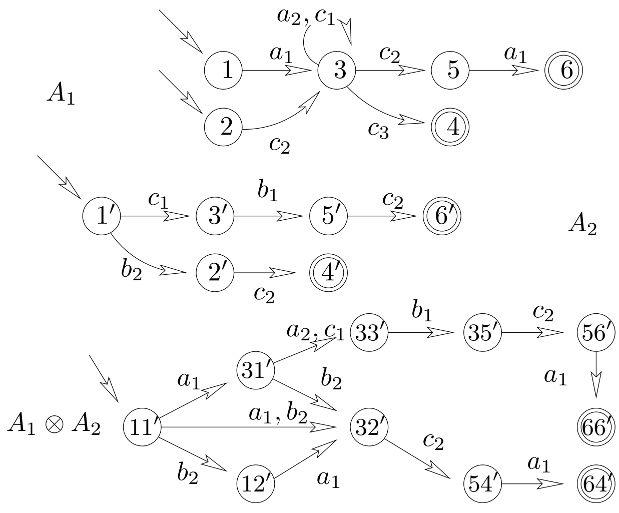

Figure 1: Two automata and their synchronisation

The synchronisation operation on two automata A_1 and A_2 builds the trim automaton where all the trajectories of both automata which cannot be synchronised according to the synchronisation events (i.e $E_1 \cap E_2$) are removed. Formally, given two automata $A_1 = (Q_1, E_1, T_1, I_1, F_1)$ and $A_2 = (Q_2, E_2, T_2, I_2, F_2)$, the *synchronisation* of A_1 and A_2, denoted $A_1 \otimes A_2$, is the trim automaton $A = \mathit{Trim}(A')$ with $A' = (Q_1 \times Q_2, E_1 \cup E_2, T', I_1 \times I_2, F_1 \times F_2)$ such that: $T' = \{((q_1, q_2), l, (q_1', q_2')) \mid \exists l_1, l_2 : (q_1, l_1, q_1') \in T_1 \wedge (q_2, l_2, q_2') \in T_2 \wedge (l_1 \cap (E_1 \cap E_2) = l_2 \cap (E_1 \cap E_2)) \wedge l = l_1 \cup l_2\}$. When $E_1 \cap E_2 = \emptyset$ (no synchronising events), the synchronisation results in the concurrent behavior of A_1 and A_2, often called the *shuffle* of A_1 and A_2. The number of states of the shuffle is the product of the number of states of A_1 and A_2 and no trajectory is removed.

Example: Figure 1 gives an example of synchronisation. The synchronising events are the c_i events.

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 longer accessible are removed from Q. Formally, given an automaton $A = (Q, E, T, I, F)$, the *restriction* of A by the states of I', denoted $A[I']$, is the automaton *Trim*(A') with $A' = (Q, E, T, I \cap I', F)$.

2.2 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. More details can be found for instance in [Pencolé and Cordier, 2005].

Definition 1 (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 which can be uncertain and are represented by an automaton, where the transition labels are observable events of E^{Mod}_{Obs}.

Definition 2 (Observation automaton)**.**
The observation automaton*, denoted Obs, is an automaton describing the observations possibly emitted by the system during the period* $[t_0, t_n]$.

The diagnosis of a discrete-event system can be seen as the computation of all the trajectories of the model that are consistent with the observations sent by the system during the period $[t_0, t_n]$. The automaton defined as the synchronisation of the model and the observations represents all these trajectories.

Definition 3 (Diagnosis)**.**
The diagnosis*, denoted* Δ*, is a trim automaton such that* $\Delta = \mathit{Mod} \otimes \mathit{Obs}$.

3 Independence properties

In this section, we present two independence properties that enable a decentralised (compact yet sufficient) representation of the diagnosis presented in the next section.

Let us first define the state-independence property which concerns the decomposition of an automaton A into two automata A_1 and A_2.

Definition 4 (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.*

Let $A' = A_1 \otimes A_2$. Note that we do not require that $A = A'$, but that A' is a super-automaton (i.e an automaton that contains all the trajectories of A and possibly more). The reason is that, when decomposing A into A_1 and A_2, the links existing between the initial states of A_1 and A_2 can be lost. We thus have $Q \subseteq Q'$, $I \subseteq I'$, and $F \subseteq F'$ where $A' = (Q', E', T', I', F')$ and $A = (Q, E, T, I, F)$.

The state-independence property is a property of a decomposition A_1 and A_2 which ensures that $A = A'$.

Definition 5 (State-independence decomposition wrt A)**.**
Two automata A_1 and A_2 are said to be a state-independent decomposition wrt A *iff $A = A_1 \otimes A_2$.*

Property 1: If A_1 and A_2 have both a unique initial state, and if they are a decomposition of A, then A_1 and A_2 are a state-independent decomposition wrt A.

Let us turn now to the transition-independence property which states that two (or more) automata do not have any transition labeled with common synchronisation events.

Definition 6 (Transition-independence)**.**
$A_1 = (Q_1, E_1, T_1, I_1, F_1)$ 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 $l \cap (E_1 \cap E_2) = \emptyset$.*

Two transition-independent automata are concurrent automata and their synchronisation is equivalent to a shuffle operation (see Section 2). Moreover, we have :

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

4 Improving diagnosis representation in a decentralised approach

In this section, we first highlight the role of state-independence property when computing diagnoses in a decentralised way. We propose then an efficient diagnosis representation based on transition-independence property.

4.1 The decentralised approach and the role of state-independence

The global model of a real-world system is too large to be explicitly built. To answer this problem, decentralised / distributed approaches have been proposed [Lamperti and Zanella, 2003; Pencolé and Cordier, 2005; Benveniste *et al.*, 2005; Jiroveanu and Boël, 2005]. In this article, we consider the decentralised approach of [Pencolé and Cordier, 2005].

The idea (see Figure 2) is to describe the system behaviour in a decomposed way by the set of its subsystems models. The so-called *decentralised* model is thus $dMod = \{Mod_1, \dots, Mod_n\}$ where Mod_i is the behavioural model of the subsystem (possibly a component) s_i. The decentralised model is designed as a decomposition of the global model *Mod*. It is generally assumed that the global model has a unique initial state and that the subsystems models also have a unique initial state. They are thus a state-independent decomposition wrt *Mod* and the global model can thus be retrieved, if needed, by $Mod = Mod_1 \otimes \dots \otimes Mod_n$.

The observations *Obs* can be decentralised as follows: $dObs = \{Obs_1, \dots, Obs_n\}$ such that Obs_i contains the observations from the subsystem s_i and such that: $Obs = Obs_1 \otimes \dots \otimes Obs_n$.

Given the local subsystem model Mod_i and the local subsystem observations Obs_i, it is possible to compute the local diagnosis $\Delta_{s_i} = Mod_i \otimes Obs_i$. These diagnoses represent the local subsystem behaviours that are consistent with the local observations. It was shown in [Pencolé and Cordier, 2005] that the set of local diagnoses is a decomposition of Δ. As they all have a unique initial state, it is also a state-independent decomposition of Δ. It is then possible to directly compute the global diagnosis of the system by synchronizing the subsystem diagnoses as follows: $\Delta = \Delta_{s_1} \otimes \dots \otimes \Delta_{s_n}$. This synchronisation operation of subsystem diagnoses is often also called a *merging* operation.

$$\begin{tikzcd}
\{Mod_1, \dots, Mod_n\} \arrow[r, "\text{local diagnosis}"] \arrow[d, "\text{synchronisation}"] & \{\Delta_{s_1}, \dots, \Delta_{s_n}\} \arrow[d, "\text{merging}"] \\
Mod \arrow[r, "\text{diagnosis}"] & \Delta
\end{tikzcd}$$

Figure 2: Principle of the decentralised diagnosis approach

It is clear that, rather than directly merging all the subsystems diagnoses together, it is possible to compute the global diagnosis by successive synchronisation operations. Let s_1 and s_2 be two disjoint subsystems (possibly being components) and let $s_{12} = s_1 \uplus s_2$ be the subsystem that contains exactly s_1 and s_2. The subsystem diagnosis $\Delta_{s_{12}}$ can be computed as $\Delta_{s_{12}} = \Delta_{s_1} \otimes \Delta_{s_2}$. The diagnosis of the system is equivalently obtained by replacing $\Delta_{s_1} \otimes \Delta_{s_2}$ by $\Delta_{s_{12}}$. We have $\Delta = \Delta_{s_{12}} \otimes \Delta_{s_3} \otimes \dots \otimes \Delta_{s_n}$.

Algorithm 1 computes the system diagnosis by starting from the component local diagnoses and merging them until ExitLoopCondition is satisfied. In this case, ExitLoopCondition checks that all pairs of diagnoses have been merged.

Algorithm 1 Computing diagnoses in a decentralised way

```
input: local diagnoses $\{\Delta_{s_1}, \dots, \Delta_{s_n}\}$
$d\Delta = \{\Delta_{s_1}, \dots, \Delta_{s_n}\}$
while $\neg$ ExitLoopCondition do
  Choose $\Delta_{s_1}, \Delta_{s_2} \in d\Delta$
  $s = s_1 \uplus s_2$
  $\Delta_s = \Delta_{s_1} \otimes \Delta_{s_2}$
  $d\Delta = (d\Delta \setminus \{\Delta_{s_1}, \Delta_{s_2}\}) \cup \{\Delta_s\}$
end while
return: $d\Delta$
```

4.2 An efficient diagnosis representation based on transition-independence

The diagnosis Δ built with a decentralised approach is equal to the synchronisation $Mod \otimes Obs$. It can be expected *Obs* sufficiently contraints the behaviours of the system to significantly reduce the size of the Δ compared to *Mod*. However Δ can still be large, especially because merging concurrent diagnoses corresponds to computing the shuffle of these diagnoses. It is why we propose to avoid these expensive shuffles by representing the system diagnosis as a set of transition-independent subsystem diagnoses.

Definition 7 (Decentralised diagnosis)**.**
A decentralised diagnosis $d\Delta$ *is a set of subsystem diagnoses $\{\Delta_{s_1}, \dots, \Delta_{s_k}\}$ such that Δ_{s_i} is the diagnosis of the subsystem s_i, $\{s_1, \dots, s_k\}$ is a partition of the system Γ, and $\forall i, j \in \{1, \dots, k\}$, $i \neq j \Rightarrow \Delta_{s_i}$ and Δ_{s_j} are transition-independent.*

As seen before, a decentralised diagnosis $d\Delta = \{\Delta_{s_1}, \dots, \Delta_{s_k}\}$ is a state-independent decomposition of the global diagnosis. We get an economical representation of the global diagnosis which can be computed, if needed, by synchronising all the subsystem diagnoses, by

$\Delta = \Delta_{s_1} \otimes \ldots \otimes \Delta_{s_k}$, which, due to Property 2, corresponds to a shuffle operation. Its final states can be obtained by a simple Cartesian product on the final states of all Δ_{s_i} (see Property 2).

Algorithm 1 can be used to compute the decentralised diagnosis from the local (component) diagnoses by changing ExitLoopCondition into $\forall \Delta_{s_1}, \Delta_{s_2} \in d\Delta$, Δ_{s_1} and Δ_{s_2} are transition-independent. Until all pairs of diagnoses are transition-independent, the algorithm chooses two transition-dependent diagnoses and merges them.

An important point to remark is the decisive role of the state-independence property of the decomposition which, combined to the transition-independence property, allows us to get an efficient representation of the system diagnosis. The state-independence property comes directly from the unique initial state property of the system and subsystems models.

5 Improving diagnosis representation in a decentralised and incremental approach

In this section, we propose to build the diagnosis on successive temporal windows, to benefit from the short duration independences. We then show that the state-independence is generally no longer satisfied, and present in the last part the abstraction operation that enables us to deal with state-dependent subsystems.

5.1 Incremental diagnosis

In the previous section, we assumed that the diagnosis was computed on a unique period. This means that the observation automaton represents the observations emitted during $[t_0, t_n]$. The problem of this approach is that long periods transition-independent behaviours are infrequent, because each component eventually interacts with most of its neighbours. Thus, all the local diagnoses will eventually be merged in a huge global diagnosis. For this reason, it can be interesting to divide the diagnosis period into temporal windows as presented in [Pencolé and Cordier, 2005], to benefit from short duration independent behaviours.

Let us consider that the diagnosis Δ_{i-1} of $[t_0, t_{i-1}]$ was built and that the set of final states of this diagnosis is F^{i-1}_{Δ}. Given the observations *Obs*^{i} of the temporal window $[t_{i-1}, t_i]$, it is possible to compute the diagnosis of this temporal window by: $\Delta^i = (\mathit{Mod}^- \otimes \mathit{Obs}^i)[F^{i-1}_{\Delta}]$ where *Mod*$^-$ is the system model *Mod* where all the states are considered as initial states. The trajectories consistent with the observations for the period $[t_0, t_i]$ (*i.e* the trajectories of Δ_i if this automaton were built) can be retrieved by the concatenation of the trajectories from Δ_{i-1} and the trajectories of Δ^i. It can also be proved that the final states of Δ_i are the final states of Δ^i. It is thus possible to incrementally add the observations of $[t_i, t_{i+1}]$ and compute the diagnosis.

5.2 Loss of the state-independence property

Our goal is to use a decentralised computation based on a decentralised model and to represent the diagnosis of the temporal window $\mathcal{W}^i = [t_{i-1}, t_i]$ by a set of transition-independent subsystem diagnoses similar to Section 4. However, contrary to Section 4, the unique initial state hypothesis can no longer be made and thus the state-independence decomposition property can no longer be considered as satisfied.

Let I^i be the set of initial states on the system for the temporal window $\mathcal{W}^i$. Given I^i, the set of initial states of a subsystem s is called the *projection* of I^i on s and denoted $I^i \downarrow s$. Let us remark that $(I^i \downarrow s) \times (I^i \downarrow s') \supseteq I^i \downarrow (s \uplus s')$. Algorithm 1 can be modified as presented in Algorithm 2 to compute a set $d\Delta^i$ of transition-independent subsystems diagnoses.

Algorithm 2 Computing transition-independent diagnoses in case of non state-independence

```
input: local diagnoses $\{\Delta_{s_1}, \ldots, \Delta_{s_n}\}$ + the set of initial states $I^i$
$d\Delta = \{\Delta_{s_1}, \ldots, \Delta_{s_n}\}$
while $\exists s_1, s_2 : \Delta_{s_1}$ and $\Delta_{s_2}$ are not transition-independent. do
    $s = s_1 \uplus s_2$
    $\Delta_s = (\Delta_{s_1} \otimes \Delta_{s_2})[I^i \downarrow s]$
    $d\Delta = (d\Delta \setminus \{\Delta_{s_1}, \Delta_{s_2}\}) \cup \{\Delta_s\}$
end while
return: $d\Delta$
```

Let us now consider the transition-independent diagnoses of $d\Delta^i$ and more precisely their final states $F^i_{\Delta_{s_j}}$. The problem is that, due to the loss of the state-independence property, F^i is only a subset of $F_{\Delta^i_{s_1}} \times \ldots \times F_{\Delta^i_{s_k}}$. Let us illustrate the problem on an example (Figure 3).

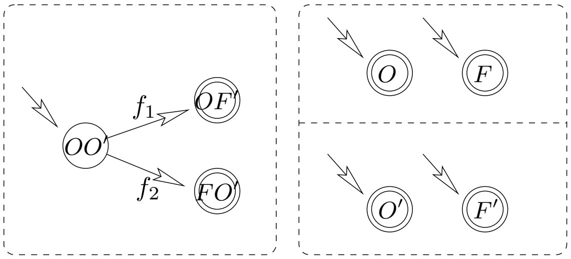

Figure 3: Example of loss of information in a naive decentralised representation of the incremental diagnosis

Example Figure 3 deals with the diagnosis of two components which can be either in a Ok state or a Faulty state. The figure presents a two-windows diagnosis, each in a box. During the first window (Figure 3, left), 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. Since the local diagnoses are transition-independent, the algorithm proposes the two local diagnoses (top and bottom in Figure 3, right) but it can be seen that the links between the initial states were lost during the projection. The decomposition of the global diagnosis is not state-independent. We have $F^2 \subset \{O, F\} \times \{O', F'\}$. To get the exact final states F^2, one solution would be to synchronise the local diagnoses in conjunction with the restriction operation on the set of final states of the first window as argument, which we want to avoid.

5.3 TI + abstract representation

To obtain the final states F^i we propose, instead of merging the state-dependent diagnoses of $d\Delta^i$, to merge their abstract representations, which is much more efficient. We thus propose to enrich the set of transition-independent diagnoses with the abstract representation of the state- and transition-independent diagnoses.

The information required relies on the initial and final states only. Thus, we define an abstraction that keeps only the information that there exists a trajectory between an initial state and a final state.

Definition 8 (Abstraction)**.**
Let $A = (Q, E, T, I, F)$ be a trim automaton. The abstraction *of A, denoted Abst(A), is the (trim) automaton $A' = (I \cup F, \{e\}, T', I, F)$ where $T' = \{(q, \{e\}, q') \mid \exists$traj $= q_0 \xrightarrow{l_1} \dots \xrightarrow{l_n} q_n \in$ Traj$(A) \wedge q_0 = q \wedge q_n = q'\}$.*

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, $(\mathit{Abst}(A_1) \otimes \mathit{Abst}(A_2))[I] = \mathit{Abst}((A_1 \otimes A_2)[I])$.

It is now possible to merge the abstract representations of the state-dependent, transition-independent diagnoses as presented in Algorithm 3 until getting a set of state- and transition independent diagnoses. This result is called the *abstract decentralised diagnosis*, and denoted $ad\Delta^i$. Given the properties 3 and 4, we have the following result:

Proposition 1.
The set of final states F^i is the set of final states of $ad\Delta^i$, i.e $\Pi_{\Delta \in ad\Delta^i} F_\Delta$.

Let us define an extended decentralised diagnosis as follows :

Definition 9 (Extended decentralised diagnosis)**.**
An extended decentralised diagnosis *on $\mathcal{W}^i$ is a pair $(d\Delta^i, ad\Delta^i)$ such that*

- *$d\Delta^i$ is the decentralised diagnosis of the system on $\mathcal{W}^i$,*
- *$ad\Delta i$ is the abstract decentralised diagnosis on $\mathcal{W}^i$.*

The transition-independent diagnoses $d\Delta^i$ provide the set of trajectories during the temporal window $\mathcal{W}^i$ and the abstract representation of state- and transition-independent diagnoses $ad\Delta^i$ provides the set of final states of the temporal window $\mathcal{W}^i$. It is thus a safe representation of the diagnosis on $\mathcal{W}^i$. It can be used as shown in 5.1 to compute the diagnosis on the whole period in an incremental way.

6 Experiments

We present the experimental results on a simplified case inspired by a real application on telecommunication networks.

Algorithm 3 Computation of the abstract decentralised diagnosis $ad\Delta^i$

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

6.1 System

The system to diagnose is a network of 14 interconnected components as presented in Figure 4.

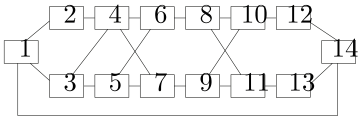

Figure 4: Topology of the network

Each component has the same behaviour: when a fault occurs, it reboots, sends a $IReboot_i$ observation and forces its neighbours to reboot by sending them a *reboot!* message. A component can also be asked to reboot by one of its neighbours. In this case, it enters into a waiting state W before rebooting and sends the $IReboot_i$ observation. In this state, it can be asked to reboot by another component (and sends then the $IReboot_i$ observation). When the reboot is finished, the component sends the observation $IAmBack_i$ and reenters the state O.

The model is presented in Figure 5. The *reboot!* message indicates that *reboot* is sent to all the neighbours, and the *reboot?* message indicates that a neighbour has sent the *reboot* message to the component. So, for example, on component 1, there are three transitions from state O to state W corresponding to a *reboot?* message received respectively from one of its three neighbours 2, 3 and 14. They are respectively labeled by $\{$*reboot*$_{2\to1}$, *IReboot*$_1\}$, $\{$*reboot*$_{3\to1}$, *IReboot*$_1\}$ and $\{$*reboot*$_{14\to1}$,*IReboot*$_1\}$.

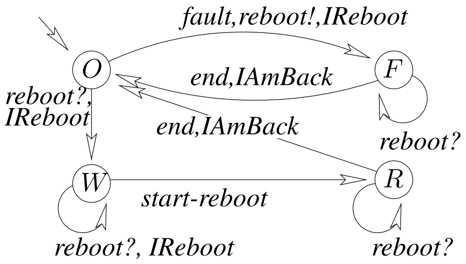

Figure 5: Model of a component

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

6.2 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 4. 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 saves us from computing the shuffle for these subsystem diagnoses, which is certainly very beneficial. However, due to the length of the window, one of the automata is still very large.
Using the method described in section 5, 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 (479) is 5% of the number of states used in the previous automaton, and the number of transitions (4 038) is less than 1% of the transitions of the previous automaton. It confirms that slicing observations is beneficial in that it increases 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. 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 slicing are merged into a unique window. The corresponding computation time is then about 3 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 interesting area for further works.

7 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. We show that the global diagnosis can be safely and economically represented by the set of diagnoses of its transition-independent subsystems. An important point is that this decentralised representation relies on the *state-independency* property.

When the period of observation is important, independent subsystems occur infrequently, since each component eventually interacts with most of its neighbours. A solution is to slice the diagnosis period into temporal windows, in order to get, on these windows, transition-independent subsystems. The problem is that the state-independency property does not hold anymore. It is then impossible to get the exact final states, at least in an economical way. On the one hand, such a set of diagnoses for transition- 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 propose in this paper 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, which produces the final states in an economic and efficient way. We show that we get a safe and efficient representation of the global diagnosis.

This work opens at least two interesting prospects. As can be seen in Algorithm 3, it is necessary to have an efficient way to check whether two abstract diagnoses are or are not state-independent, and we are currently working on this point. Another concern is about the slicing. As shown in section 6, 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.

Acknowledgements

A. Grastien is currently supported by National ICT Australia (NICTA) in the framework of the SuperCom project. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

References

[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.

[Jiroveanu and Boël, 2005] G. Jiroveanu and R. Boël. Petri net model-based distributed diagnosis for large interacting systems. In *Sixteenth International Workshop on Principles of Diagnosis (DX-05)*, pages 25–30, 2005.

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

[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.

[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.