A Spectrum of Symbolic On-line Diagnosis Approaches*

Anika Schumann
National ICT Australia &
The Australian National University
Canberra ACT 0200, Australia
Anika.Schumann@anu.edu.au

Yannick Pencolé
LAAS-CNRS, University of Toulouse
7 avenue du Colonel Roche
31000 Toulouse, France
Yannick.Pencole@laas.fr

Sylvie Thiébaux
National ICT Australia &
The Australian National University
Canberra ACT 0200, Australia
Sylvie.Thiebaux@anu.edu.au

Abstract

This paper deals with the monitoring and diagnosis of large discrete-event systems. The problem is to determine, on-line, all faults and states that explain the flow of observations. Model-based diagnosis approaches that first compile the diagnosis information off-line suffer from space explosion, and those that operate on-line without any prior compilation have poor time performance. Our contribution is a broader spectrum of approaches that suits applications with diverse time and space requirements. Approaches on this spectrum differ in the amount of reasoning and compilation performed off-line and therefore in the way they resolve the tradeoff between the space occupied by the compiled information and the time taken to produce a diagnosis. We tackle the space and time complexity of diagnosis by encoding all approaches in a symbolic framework based on binary decision diagrams. This allows for the compact representation of the compiled diagnosis information, and for its handling across many states at once rather than for each state individually. Our experiments demonstrate the diversity and scalability of our symbolic methods spectrum, as well as its superiority over the corresponding enumerative implementations.

Introduction

There is an increasing need for automated monitoring and supervision tools for large discrete-event systems in areas as diverse as telecommunication, power distribution, manufacturing, spatial exploration, and web services. Such tools aim at assisting the operator in charge of the system supervision with tasks that include diagnosis, reconfiguration, and control.

This paper is concerned with automated diagnosis, and more specifically with the on-line identification of the faults that explain the continual flow of observations received from the system. Existing model-based approaches typically fall into two categories. In the first, a significant amount of off-line reasoning is performed to compile the system model into a larger model that embeds diagnosis information. This information, generated once and for all, is then exploited on-line to more efficiently produce the diagnosis from the actual observations. In the second category, no such compilation is performed and all the reasoning is done on-line.

The diagnoser approach (Sampath *et al.* 1996) is the archetype of compilation-based techniques. Off-line, it compiles all possible diagnoses into a finite-state machine (the diagnoser). On-line, this machine is simply run to efficiently retrieve the diagnoses explaining the current flow of observations. Unfortunately, diagnosers can be so large that they are not computable for all but the smallest applications. On-line simulation based approaches (Baroni *et al.* 1999) fall in the no-compilation camp. They directly compute the diagnosis from the behavioral model of the system by simulating possible trajectories. Here the space requirements are reasonable, but the simulation time can be excessive for large applications.

Clearly we need a more flexible resolution of the tradeoff between on-line and off-line computation, that is, between time and space. Research in that direction includes the decentralised diagnoser approach (Pencolé & Cordier 2005) which precomputes diagnosers for small subsystems only, but needs to ensure consistency of the local diagnoses at run-time. Another recent line of work deals with the incremental on-line compilation of diagnosis information and its reuse (Lamperti & Zanella 2006).

In this paper, we take an orthogonal approach to resolving the space-time tradeoff. We present a spectrum[1] of methods which differ by the degree of reasoning performed off-line and by the nature and the size of the underlying compiled models. These methods range from no compilation to full compilation of diagnosis information, but are not limited to those extreme cases.

To increase efficiency, all models are represented by symbolic finite-state machines using binary decision diagrams (BDDs), and all methods are implemented via symbolic operations. BDDs enable the compact encoding and the implicit manipulation of sets of states and transitions. On the one hand, they allow us to reduce the space requirements of models with a high degree of compilation. On the other hand, they help reducing the diagnosis time of approaches with a low degree of compilation by avoiding the individual consideration of all possible diagnosis explanations.

Our experiments illustrate the diversity of space-time requirements of methods across the spectrum, and clearly demonstrate the superiority of our symbolic methods over the equivalent enumerative ones.

*This work was supported by NICTA's SuperCom project. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part via the ARC.

[1]This is not to be confused with the spectrum of diagnosis definitions presented in (Brusoni *et al.* 1998) nor the spectrum of symbolic compilations in (Darwiche & Marquis 2002).

The paper is organised as follows. After a brief reminder of BDDs and symbolic finite state machines, we present the successive models underlying the respective methods, give an on-line diagnosis algorithm for each of them, experimentally illustrate the strength of our approach, and conclude with related and future work.

Symbolic Finite State Machines

Ordered binary decision diagrams (OBDDs, or BDDs for short) (Bryant 1986) are a form of reduced decision graph that provide a compact canonical representation of boolean functions $\mathcal{B}^n \mapsto \mathcal{B}$. While the BDD representation still requires exponential space in the number of boolean variables in the worst case, the reductions often make the BDD of a function much smaller than its disjunctive normal form (DNF). Any boolean operation $f \star g$ on two BDDs f and g, can be carried out in $\mathbb{O}(|f||g|)$ at most, where $|f|$ denotes the number of nodes in the BDD f.

In our approach, the finite-state machines (FSMs) describing our diagnosis models are encoded symbolically, by means of BDDs, and all diagnosis algorithms are implemented in terms of BDD operations. This confers us the ability to compactly represent and efficiently manipulate sets of states and transitions.

To encode the set of states X and the set of events Σ of a FSM, it is necessary to introduce $Nr(Q) = \lceil \log_2 |Q| \rceil$ boolean variables for each set Q. Thus the events labelling the transitions can be encoded with the boolean variables $b^\Sigma = \{b_1^\Sigma, \ldots, b_{Nr(\Sigma)}^\Sigma\}$ and the states with the variables $b^X = \{b_1^X, \ldots, b_{Nr(X)}^X\}$. The initial state of the FSM is then simply given by a boolean function (represented by a BDD) over these state variables. For instance, in a 6 state FSM, the state x_2 would be given by the conjunction $\neg b_3^X \wedge b_2^X \wedge \neg b_1^X$, and the set of states $\{x_2, x_5\}$ by the DNF $(\neg b_3^X \wedge b_2^X \wedge \neg b_1^X) \vee (b_3^X \wedge \neg b_2^X \wedge b_1^X)$.

Transitions require the introduction of another set of state variables $b^{X'} = \{b_1^{X'}, \ldots, b_{Nr(X)}^{X'}\}$, called the primed variables, which are used to represent the target states of the transitions. Each transition can then be given as a conjunction involving the state variables, event variables, and primed variables. For instance, in a FSM consisting of 6 states and 3 events, the transition $t = x_2 \xrightarrow{\sigma_1} x_5$ would be given by $t = (\neg b_3^X \wedge b_2^X \wedge \neg b_1^X) \wedge (\neg b_2^\Sigma \wedge b_1^\Sigma) \wedge (b_3^{X'} \wedge \neg b_2^{X'} \wedge b_1^{X'})$. The transition relation, i.e, a set T of transitions, can then be given as a DNF which the BDD data structure will hopefully greatly reduce.

Spectrum of Symbolic Diagnosis Models

This section formally defines four symbolic models on which we base our diagnosis algorithms. These models are inspired from (Schumann, Pencolé, & Thiébaux 2004). Rather than merely using them as successive steps in the computation of a symbolic diagnoser, as Schumann *et al.* do, we adapt them and build efficient *on-line* diagnosis algorithms upon each of them. These algorithms and their evaluation are the main technical contributions of the paper.

The models differ in the extent to which information is compiled, starting with the component models, the simple representation of the system without any precomputation, to the diagnoser model which compiles the diagnosis information for every possible sequence of observations. We choose the encodings that make use of BDDs as few as possible while still allowing an efficient on-line retrieval of diagnosis information. Efficiency requires for instance that we partition the transition sets of our FSMs.

Since the focus of the paper is the use of the models for on-line diagnosis, we will only briefly allude to their off-line computation. We refer the reader to (Schumann, Pencolé, & Thiébaux 2004) for details of how this might be done.

Component Models

As in (Sampath *et al.* 1996), the diagnosed system is composed of a set of n individual components G_i, with respective sets of states X_i, and a global event set Σ. The events are partitioned into observable Σ_o and unobservable Σ_u events, the latter of which is further partitioned into faults Σ_f and shared Σ_s events. The shared events are used to describe the communication between components.

Following the usual symbolic FSM representation described above, the symbolic components are

$$G_i = \langle b^{X_i}, b^{X_i'}, b^\Sigma, x_{0_i}, T_{o_i}, T_{s_i}, T_{f_i} \rangle,$$

where b^{X_i}, $b^{X_i'}$, and b^Σ are the Boolean variables that define the following BDDs: x_{0_i} to represent the initial state and $T_{o_i}, T_{s_i}, T_{f_i}$ to represent the observable, shared and fault transitions. Note that every transition in every component G_i is defined over the same global event variables but over local state variables, that is, over $b^{X_i} \cup b^\Sigma \cup b^{X_i'}$.

Global Model

Diagnosing directly from the component models, without any compilation at all, is space efficient but very slow. Our second model incorporates a limited form of compilation arising from performing synchronisation off-line.

The global model is the synchronous product of the n component models: its state space is the Cartesian product of the state spaces of the components and its transitions are synchronised in that any shared event always occurs simultaneously in all components that define it. Similarly to the component models it is symbolically represented as

$$G = \langle b^X, b^{X'}, b^\Sigma, x_0, T_o, T_s, T_f \rangle,$$

where $b^X = \cup_{i=1}^n b^{X_i}$ (resp. $b^{X'} = \cup_{i=1}^n b^{X_i'}$) is the union of the components local state (resp. primed) variables. State $x_0 = \wedge x_{0_i}$ is the initial state. Also the BDDs T_o, T_s, T_f representing the global observable, shared and fault transitions are computed from the local transitions mainly by applying the $\wedge$ operator.

Abstracted Model

Diagnosing based on the global model is also not very efficient, since only limited information about unobservable events has been compiled away. We therefore add another model to our spectrum, the abstracted model, which is derived from the global one by abstracting all unobservable non-fault transitions and the order in which faults can occur.

Hence its states $\widetilde{X}$ are obtained from the global ones, by removing all states (except the initial one) that are not the start or target state of an observable transition T_o. All sequences of unobservable events are replaced by a single transition labelled with the union of the corresponding faults (which can be empty if the sequence consists only of shared events). The set $\widetilde{T}_f$ of these new transitions is defined as

$$\Big\{x \xrightarrow{l} x' \mid \exists \text{ path } x \xrightarrow{\sigma_1} x_1 \cdots \xrightarrow{\sigma_{k-1}} x_{k-1} \xrightarrow{\sigma_k} x' \text{ in } G \text{ with } x, x' \in \widetilde{X}, \sigma_1, \ldots, \sigma_k \in \Sigma_u \text{ and } l = \{\sigma_1, \ldots, \sigma_k\} \cap \Sigma_f\Big\}.$$

Figure 1 shows an example of an abstracted model. In the symbolic setting, the abstracted model

$$\widetilde{G} = \langle b^X, b^{X'}, b^{\Sigma_o}, b^F, x_0, T_o, \widetilde{T}_F\rangle$$

is encoded using the same boolean state variables as the global model, the subset b^{Σ_o} of boolean variables representing the observable events, and an additional $|\Sigma_f|$ variables $b^F = \{b_1^f, \ldots, b_{|\Sigma_f|}^f\}$ needed for the fault transition labels $F \subseteq 2^{\Sigma_f}$. There is a one to one correspondence between fault events and these variables, and a fault transition label is encoded as a conjunction of literals over b^F whose signs depend on whether the corresponding fault belongs to the label. Note that the abstracted states $\widetilde{X} \subseteq X$ are encoded over the same boolean variables as the global ones, since their number is not significantly smaller than $|X|$.

Diagnoser Model

The abstracted model still requires the on-line computation of fault information, which slows down on-line diagnosis. We therefore also consider a diagnoser model in which this entire information is compiled. A diagnoser is a deterministic finite state machine whose transitions are only labelled with observable events and whose states are directly labelled by the diagnosis information that is consistent with the past observations. This information consists of a sets of pairs (x, l) denoting a state and a fault label of the abstracted model. Let $\hat{X}$ be the set of diagnoser states, and let $\hat{x}_0$ be the initial diagnoser state. Let $\hat{R}$ denote the diagnoser state labelling function which associates a diagnoser state to the pairs in its label and verifies $\hat{R}(\hat{x}_0) = \{(x_0, \emptyset)\}$. The set $\hat{T}$ of diagnoser transitions then satisfies: $\hat{x} \xrightarrow{\sigma} \hat{x}' \in \hat{T}$ iff

$$\hat{R}(\hat{x}') = \Big\{ (x', l') \mid \exists (x, l) \in \hat{R}(\hat{x}) \text{ such that } \exists (x \xrightarrow{l''} x'') \in \widetilde{T}_F \text{ and } \exists (x'' \xrightarrow{\sigma} x') \in T_o \text{ and } l' = l \cup l'' \Big\}.$$

Figure 1 gives an example. Symbolically the diagnoser

$$\hat{G} = \langle b^{\hat{X}}, b^{\hat{X}'}, b^X, b^{\Sigma_o}, b^F, \hat{x}_0, \Phi, \hat{T}\rangle$$

is encoded using the additional variables $b^{\hat{X}}$ and $b^{\hat{X}'}$ for representing diagnoser states in their role as start and target states of transitions. The BDD Φ encodes the diagnoser state labelling function $\hat{R}$ and is defined over the variables $b^{\hat{X}} \cup b^X \cup b^F$.

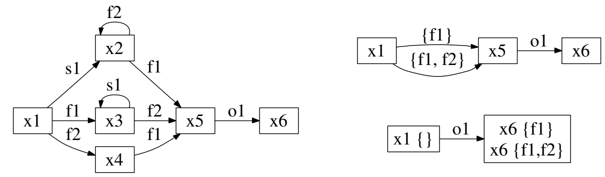

Figure 1: Global (left), abstracted (top right) and diagnoser models (bottom right). $\Sigma_o = \{o_1\}, \Sigma_s = \{s_1\}, \Sigma_f = \{f_1, f_2\}$

Symbolic On-line Diagnosis

On-line diagnosis aims to detect faults while the system is working. Given a sequence of observations, it identifies all the faults and system states that are consistent with the occurrence of these events. For each of the above models, we give a procedure that uses symbolic reasoning to compute this diagnosis information as efficiently as possible.

Initially the system is in state x_0 and no fault has occurred, so the diagnosis information is $x_0 \wedge F_\emptyset$, where $F_\emptyset = \bigwedge_{j=1}^{|\Sigma_f|} \neg b_j^f$ denotes the empty fault label. Now, each time an event σ is observed, the diagnosis information $\hat{x}'_{info}$ is derived based on σ, one of the models, and the previous diagnosis information $\hat{x}_{info}$. In this section we show how we can symbolically retrieve $\hat{x}'_{info}$ using the basic boolean operations and the following ones:

- IsDef(bdd) returns true iff bdd does not represent *false*,
- Extract(bdd, B)
  deletes from bdd all occurrences of variables not in B,
- Abstract(bdd, B)
  deletes from bdd all occurrences of variables in B,
- Swap(bdd, $\{a_1, \ldots, a_k\}, \{b_1, \ldots, b_k\}$) renames, in bdd, variable a_i with b_i, $i = 1 \ldots k$, and vice versa.

For the sake of readability, the algorithms are presented in the following order from the diagnoser to the component based one.

On-line diagnosis based on the diagnoser

The precomputed diagnoser contains all the information to perform efficient on-line diagnosis. Given the previous diagnoser state $\hat{x}$ and a new observation σ it is sufficient to

1. trigger the corresponding transition and
2. retrieve the fault information from its target state.

Algorithm 1 describes the symbolic procedure. To trigger the transition (step 1), we apply three BDD operations, namely $\wedge$, $Extract$, and $Swap$. Applying the $\wedge$ operation to the encodings of a start state $\hat{x}$, an event σ and the transition set $\hat{T}$, retrieves the transition that starts in $\hat{x}$ and is labelled with σ. Next, the operation $Extract$ is used to obtain only the target state $\hat{x}'$ of the transition. We then $Swap$ the encoding of $\hat{x}'$ over the primed variables for an encoding over the non-primed ones in order to determine its label and to trigger future transitions. To determine the label of $\hat{x}'$ (step 2), we first conjoin $\hat{x}'$ with the state labelling function Φ, and then abstract from the boolean variables representing diagnoser states.

Algorithm 1 $DiagDiagnose(\hat{G}, \hat{x}, \sigma)$

1: $\hat{x} \leftarrow \hat{x} \wedge \sigma \wedge \hat{T}$
   $\hat{x}' \leftarrow \text{Extract}(\hat{x}, b^{\hat{X}'})$
   $\hat{x}' \leftarrow \text{Swap}(\hat{x}', b^{\hat{X}}, b^{\hat{X}'})$
2: $\hat{x}'_{info} \leftarrow \hat{x}' \wedge \Phi$
   return Abstract $(\hat{x}'_{info}, b^{\hat{X}})$

On-line diagnosis based on the abstracted model

Using the abstracted model, the retrieval of the diagnosis information given its predecessor $\hat{x}_{info}$ requires:

1. computing states $\widetilde{X}_{unObs}$ that can be reached from those contained in $\hat{x}_{info}$ before observing the new event σ,
2. computing the fault labels representing the faults that have occurred on a path from the initial state to a state in $\widetilde{X}_{unObs}$, and
3. triggering all transitions starting from states in $\widetilde{X}_{unObs}$ and labelled σ.

The corresponding three symbolic computation steps are shown in Algorithm 2. Once the unobservable transitions $\widetilde{T}_{unObs}$ starting in a state of $\hat{x}_{info}$ are determined (line 1), they contain all the new faults that could have occurred since the last observation. These are added to the faults in $\hat{x}_{info}$ that have previously occurred using function $AddFault$. Symbolically, adding a fault f_i implies changing the value of the corresponding boolean variable b_i^f from false to true. It is done by abstracting b_i^f from the fault label l (i.e. $Abstract(l, \{b_i^f\})$) and conjoining it with l (i.e. $l \wedge b_i^f$). This abstraction can be done simultaneously for all fault labels L_{f_i} to which f_i has to be added.

Finally the observable transitions are triggered and the new diagnosis information returned (step 3).

Algorithm 2 $AbstDiagnose(\widetilde{G}, \hat{x}_{info}, \sigma)$

1: $\widetilde{T}_{unObs} \leftarrow \widetilde{T}_f \wedge Extract(\hat{x}_{info}, b^X)$
2: $\widetilde{X}_{unObs} \leftarrow \hat{x}_{info} \vee AddFault(\widetilde{T}_{unObs}, \hat{x}_{info})$
   $\widetilde{X}_{unObs} \leftarrow Swap(\widetilde{X}_{unObs}, b^X, b^{X'})$
3: $\hat{x}'_{info} \leftarrow Extract(\widetilde{X}_{unObs} \wedge \sigma \wedge T_o, b^{X'} \cup b^F)$
   return $Swap(\hat{x}'_{info}, b^X, b^{X'})$

On-line diagnosis based on the global model

Using the global model, the symbolic computation of the diagnosis information is similar to that above, except that states $\widetilde{X}_{unObs}$ now need to be computed based on transition sequences in G. For this purpose, we first combine shared and fault transitions into a single transition set T_u, in which all events are defined over variables b^F. Here shared transitions are labelled with the empty fault label $F_\emptyset$.

Algorithm 3 describes the symbolic procedure. All formulas $\widetilde{X}_{unObs}$, X_{new}, and X_{targ} represent sets of labelled states, that is, sets of tuples (x, l). $\widetilde{X}_{unObs}$ is computed using breadth-first search (lines 1-8). Initially $\widetilde{X}_{unObs}$ and X_{new} are composed of the previous diagnosis information $\hat{x}_{info}$ (lines 1-2). As long as there are still new diagnosis tuples X_{new} that have not been processed (line 3), applicable unobservable transitions are triggered (line 4) and any fault labelling them is added (line 5). The tuples already closed are removed from the resulting tuples X_{targ} to ensure the termination of the algorithm (operator $\wedge\neg$ in line 6). The new tuples are added to the set of closed ones (operator $\vee$ in line 7). Once $\widetilde{X}_{unObs}$ is obtained, the new diagnosis information is retrieved as in step 3 of Algorithm 2 (line 9).

Algorithm 3 $GlobDiagnose(T_u, T_o, \hat{x}_{info}, \sigma)$

1: $X_{new} \leftarrow \hat{x}_{info}$
2: $\widetilde{X}_{unObs} \leftarrow X_{new}$
3: **while** $IsDef(X_{new})$ **do**
4:    $T_{new} \leftarrow T_u \wedge Extract(X_{new}, b^X)$
5:    $X_{targ} \leftarrow AddFault(T_{new}, X_{new})$
      $X_{targ} \leftarrow Swap(X_{targ}, b^X, b^{X'})$
6:    $X_{new} \leftarrow X_{targ} \wedge \neg\widetilde{X}_{unObs}$
7:    $\widetilde{X}_{unObs} \leftarrow \widetilde{X}_{unObs} \vee X_{new}$
8: **end while**
9: $\hat{x}'_{info} \leftarrow Extract(\widetilde{X}_{unObs} \wedge \sigma \wedge T_o, b^{X'} \cup b^F)$
   return $Swap(\hat{x}'_{info}, b^X, b^{X'})$

On-line diagnosis based on the component models

In addition to the previous algorithm, on-line diagnosis based on the component models requires the computation of those of the global transitions that are needed to determine the new diagnosis information. For every component G_i we only need to consider

- all sequences of unobservable transitions T_{u_i} starting in a state $x_i \in X'_i$ consistent with the previous diagnosis information ($X'_i = Extract(\hat{x}_{info}, b_i^X)$) and
- all observable transitions T_{σ_i} labelled with the new observation σ ($T_{\sigma_i} = \sigma \wedge T_{o_i}$).

To obtain the corresponding global transitions efficiently, via the $\wedge$ operator, a synchronous product is required. In a synchronous system, when a transition is triggered in a component G_i, a transition is also triggered in every other component. Hence we add for every event σ' that can occur in G but not in G_i and every state x of a component model G_i, a transition $x \xrightarrow{\sigma'} x$. Now the relevant global transitions are computed as follows:

- $T_u \leftarrow \wedge_{i=1}^n T_{u_i}$ and similarly
- $T_\sigma \leftarrow \wedge_{i=1}^n T_{\sigma_i}$.

Using these two transition sets, the new diagnosis information is computed as in Algorithm 3. The only change needed is the replacement of $\sigma \wedge T_o$ with T_σ in line 9 of the algorithm.

Experimental Evaluation

We implemented our approach on top of the CUDD BDD package (`http://vlsi.colorado.edu/~fabio/`

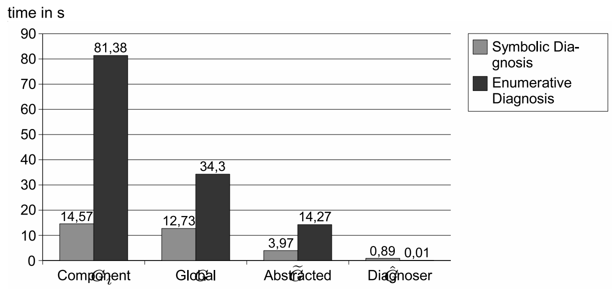

Figure 2: Average diagnosis times over 100 scenarios of 10000 observations each.

CUDD). In order to evaluate the benefits of our symbolic framework, we also implemented a traditional "enumerative" version of the models and algorithms, using optimised automata data structures which facilitate the manipulation of individual states and transitions. These two implementations enable us to present experimental evidence that the symbolic approach yields important gains in time or space.

Our experiments below were run on a 1.2 GHz Pentium IV with 512 Mb of memory. We first use the largest example in (Schumann, Pencolé, & Thiébaux 2004), which is derived from a telecommunication application. It consists of 3 components (a switch with 12 states and 18 transitions, a primary control station of 13 states and 15 transitions, and a backup control station of 19 states and 28 transitions), 9 observable events, 11 fault types, and 8 other unobservable events. We generated by simulation 100 arbitrary scenarios (possible sequences of observations) of 10000 observations each, and used them as input to all models.

Figure 2 compares the time performance of the various on-line diagnosis methods. All symbolic models except the diagnoser are more efficient to use than their enumerative counterparts. This should not come as a surprise: BDDs are well suited to triggering transition sets and enable the consideration of all diagnosis tuples at once, but do not generally pay off when only a single transition is involved as is the case with the diagnoser.

The differences in symbolic diagnosis times across the spectrum correlate with the extent to which the accumulation of faults (function $AddFault$ described on page 4) is performed on-line. Even though a fault f_i can be simultaneously added to all fault labels L_{f_i}, $AddFault$ still requires the individual consideration of fault labels (in general, $L_{f_i} \neq L_{f_j}$ for $f_i \neq f_j$), which is the main bottleneck of the symbolic computation. The component and global models yield similar diagnosis times because $AddFault$ is applied the same number of times in both cases and symbolic synchronization is very fast. In contrast, the abstracted model yields significantly faster diagnosis times because $AddFault$ only needs to be applied once per observation. With the diagnoser, $AddFault$ is never called.

Taken in conjunction with the diagnosis times, the corresponding model sizes (see Table 1) illustrate the time/space tradeoff of the methods across the spectrum and the superiority of the symbolic approach. Comparing the symbolic models (resp. the enumerative ones), we can state, that the faster the on-line diagnosis based on a model, the larger the

	G_i	G	$\widetilde{G}$	$\hat{G}$
states Nr.	∅ 17.7	1063	965	18474
transition Nr.	∅ 34	2912	48958	120698
space symb. (Mb)	0.01	0.2	0.6	7.5
space enum. (Mb)	0.01	0.2	2.7	123.9

Table 1: Model sizes

model size. For all models, the symbolic representation is as small as or smaller than the enumerative one; yet except for the diagnoser, the symbolic run-times are significantly better. Importantly, the symbolic diagnoser is as small as $\frac{1}{20}$ the size of the enumerative one. Its size is rather comparable to that of the enumerative abstracted model, yet it is an order of magnitude faster than the latter.

Focusing on the symbolic spectrum, the abstracted model appears to provide a particularly interesting tradeoff. It is 13 times smaller than the diagnoser but only 4 times slower. Compared to the global model, the percentage decrease of diagnosis time of the abstracted model is slightly higher than its percentage increase in size. The advantage of the abstracted model results from the efficiency of (1) the symbolic triggering of sets of transitions, and (2) the update of fault labels by considering fault sets rather than by considering a sequence of individual faults.

The component model also presents an interesting tradeoff due to its very small size of only 8 kilobytes. For large applications, it appears to be the only option. We show how the component-based approach scales as the size of the system increases, using a grid of computer nodes inspired from the example in (Rintanen & Grastien 2007). All nodes have the same behaviour. In normal mode, each node performs its task, sending an *on* message to a supervisor prior to starting and an *off* message upon completion. When a node becomes faulty, an automatic recovery system forces the node to reboot and to send his neighbours reboot requests which get propagated through the grid.

The model of a node has 14 states, 67 transitions, 1 fault, 8 shared, and 2 observable events. The global model of a grid of size $n \times m$ closely approaches the $14^{m \times n}$ states bound. E.g., the 2×2 grid has $14^{3.85} \simeq 26,000$ states. The example is poorly diagnosable. Every system state can be associated with the 2^{m*n} fault hypotheses, and the observations do not allow discrimination between faults due to a masking phenomenon (nodes reboot silently and reboot requests from other nodes are not observed). Consequently, there is a huge set of diagnoses that explains a given observation sequence.

Figure 3 compares the performance of the symbolic and enumerative approaches as the size of the grid increases (note the logarithmic scale). The gap between the two approaches increases by an order of magnitude with each addition of a new component. For the three larger grids, the enumerative approach failed to refine the diagnosis within an average of 10 sec – the theoretical number of diagnoses for the 2×2 grid is 2 458 624. All other enumerative approaches are unsuitable, as the enumerative global model could not be computed. In contrast, the symbolic approach was able to refine the same diagnosis in 0.079 sec.

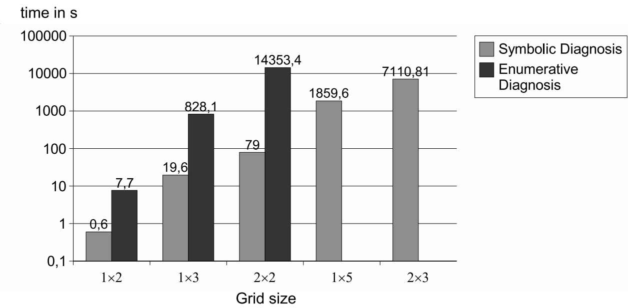

Figure 3: Average component-based diagnosis times over 10 scenarios of 1000 observations each

Conclusion, Related & Future Work

We have presented a spectrum of symbolic diagnosis approaches which differ in the amount of model compilation performed off-line. The underlying models range from the small component models that do not incorporate any compilation, to the diagnoser model in which the diagnosis information is compiled for the entire observable behaviour of the system. The abstracted model constitutes an interesting alternative to the diagnoser: it is considerably smaller but not much slower and so applies to a wider range of applications.

Thanks to the symbolic implementation, we are able to handle large sets of transitions and diagnosis hypotheses at once. This leads to a simple and efficient way of obtaining the correct and complete set of diagnoses. In comparison to an enumerative implementation, only the on-line use of the symbolic diagnoser incurs a small time overhead. In all other cases the run-time of the symbolic approach is significantly reduced, and so are the space requirements of the larger models. Therefore, an enumerative approach is mainly useful for very small applications for which the computation and storage of the large diagnoser is feasible.

There are only few other works presenting results obtained with generic on-line diagnosis software. The UMDES Library (`http://www.eecs.umich.edu/umdes/`) provides an enumerative implementation of Sampath's diagnoser (Sampath *et al.* 1996). UMDES cannot compete with either our symbolic or enumerative implementations. In fact, one of our motivations to implement our own enumerative algorithms was that UMDES was unable to compute the diagnoser for the smallest of the examples given in (Schumann, Pencolé, & Thiébaux 2004).

The idea of exploiting symbolic representations in the context of discrete-event systems diagnosis is not new but it has traditionally been applied to different problems, e.g. checking diagnosability (Cimatti, Pecheur, & Cavada 2003; Rintanen & Grastien 2007), off-line diagnosis using off-the-shelf model-checkers (Cordier & Largouët 2001), or computing a symbolic diagnoser (Marchand & Rozé 2002; Schumann, Pencolé, & Thiébaux 2004). In contrast, we exploit the power of the symbolic representation to design a *range* of efficient *on-line* diagnosis approaches.

Symbolic representations based on Decomposition Negation Normal Forms (DNNFs) have successfully been applied to diagnosing *static* systems (see e.g., (Darwiche 1998)). For diagnosing *dynamic* systems however, BDDs are better suited because the main operation, namely the triggering of transitions, can be performed in polynomial time in the size of the BDD while it would require exponential time in the size of the DNNF.

In (Pencolé & Cordier 2005) the authors resolve the time/space complexity tradeoff using a single approach which merges, on-line, the results of a set of diagnosers compiled for small subsystems. We plan to extend our symbolic spectrum to decentralised approaches, such as this one.

Another line of future work is to extend our framework to stochastic systems and compute probability distributions on diagnoses, using for instance algebraic decision diagrams which are generalisation of BDDs to real-valued functions over the booleans. Finally, integrating diagnosis and planning for repair or reconfiguration actions is one of the most significant challenges faced by the field of model-based diagnosis (Console & Dressler 1999). Given the recent success of planning techniques based on symbolic model-checking, we believe that our framework will prove a good basis for addressing this challenge.

References

Baroni, P.; Lamperti, G.; Pogliano, P.; and Zanella, M. 1999. Diagnosis of large active systems. *Artificial Intelligence* 110(1):135–183.

Brusoni, V.; Console, L.; Terenziani, P.; and Dupre, D. T. 1998. A spectrum of definitions for temporal model-based diagnosis. *Artificial Intelligence* 102(1):39–79.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Computers* C-35(8):677–691.

Cimatti, A.; Pecheur, C.; and Cavada, R. 2003. Formal verification of diagnosability via symbolic model checking. In *Proc. IJCAI-03*, 363–369.

Console, L., and Dressler, O. 1999. Model-based diagnosis in the real world: lessons learned and challenges remaining. In *Proc. IJCAI-99*.

Cordier, M.-O., and Largouët, C. 2001. Using model-checking techniques for diagnosing discrete-event systems. In *Proc. DX-01*, 39–46.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *JAIR* 17:229–264.

Darwiche, A. 1998. Model-based diagnosis using structured system descriptions. *JAIR* 8:165–222.

Lamperti, G., and Zanella, M. 2006. Flexible diagnosis of discrete-event systems by similarity-based reasoning techniques. *Artificial Intelligence* 170:232–297.

Marchand, H., and Rozé, L. 2002. Diagnostic de pannes sur des systèmes à événements discrets: une approche à base de modèles symboliques. In *13ème Congrès AFRIF-AFIA de Reconnaissances des Formes et Intelligence Artificielle*, 191–200.

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

Rintanen, J., and Grastien, A. 2007. Diagnosability testing with satisfiability algorithms. In *Proc. IJCAI-07*, 532–537.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; and Teneketzis, D. 1996. Failure diagnosis using discrete event models. *IEEE Trans. on Control Systems Techn.* 4(2):105–124.

Schumann, A.; Pencolé, Y.; and Thiébaux, S. 2004. Diagnosis of discrete-event systems using binary decision diagrams. In *Proc. DX-04*, 197–202.