Presented at the 1998 IEEE International Conference on Systems, Man, and Cybernetics,
Hyatt Regency La Jolla, San Diego, California, USA , October 11-14, 1998.

A Generic and Symbolic Model-Based Diagnostic Reasoner with Highly Scalable Properties

Amit Misra, Gregory Provan, Gabor Karsai, George Bloor, Ethan Scarl

Abstract Modern computing technologies - hardware, software, and algorithmic – have enabled the deployment of more exacting diagnostic reasoning (DR) systems than has heretofore been possible. Compromises in algorithm and modeling paradigm complexity, due to computational throughput and state-space explosion constraints, have historically dominated practical applications of such systems. This paper describes approaches that have been shown to be applicable in a wide set of domains. The algorithms used are highly scaleable and support a symbolic modeling formalism for analyzing the properties of the complex, dynamic systems. Moreover, analysis of simultaneous failures occurs as a natural byproduct of this formalism.

Index Terms—Diagnostics, causal network, OBDD.

CHALLENGES OF DEVELOPING DIAGNOSTICS SYSTEMS FOR COMPLEX APPLICATIONS

There exists a wide variety of modeling and inference methods for diagnostics, including expert systems, neural networks, and model-based approaches. All of these approaches can represent and analyze the same artifact but from different viewpoints, and with different fidelity and cost

Complex systems, such as commercial aircraft, represent a challenge both in modeling and inference. Next generation improvements in the performance of diagnostic systems will require, at the very least, higher fidelity models, guarantees on model consistency, and reasoner algorithms that are less susceptible to state-space-explosion problems.

Moreover, analyses of the diagnostic models for complex systems must be exacting, and exhaustive. Incomplete analyses using probabilistic arguments become progressively more tenuous for complex systems and unacceptable for safety critical applications.

Powerful and generic modeling paradigms and algorithms that seamlessly serve many analytic needs are key ingredients for developing the next generation diagnostic systems. Models that capture temporal and dynamic system aspects whose representations scale to complex and non-deterministic scenarios are critical.

To address these challenges in designing diagnostic systems for complex applications, we propose a model-based solution that integrates two state-of-the-art technologies: (1) causal network diagnostic modeling and inference algorithms and (2) Ordered Binary Decision Diagram-based (OBDD) inference algorithms.

Taking advantage of the efficient symbolic manipulations using OBDDs, researchers have solved a wide range of problems in hardware verification, testing, real-time systems, and mathematical logic that would have been otherwise impossible due to combinatorial explosion. Symbolic model checking is extensively used in hardware design (see, e.g., [11]), and has shown to be efficient in state space sizes 10^{120} and beyond.

We believe these two technologies can synergistically provide:

- A generalized modeling and inference approach to failure domain problems, e.g., diagnostics, safety, and reliability.
- A symbolic formalism (OBDD) that enables a natural and mathematical analysis of simultaneous failures.
- A significantly enhanced ability to scale to large complex applications while maintaining accuracy.
- The ability to guarantee bounds on computational requirements, *a priori*, based on modeling paradigm fidelity.

In this paper the first section summarizes the diagnostic representation and inference algorithms known as causal networks [2]. The second section summarizes the diagnostic representation and inference algorithms based on Ordered Binary Decision Diagrams [3,4,5]. The final section of the document outlines the approach to integrating these two technologies and overviews the long-term vision.

CAUSAL NETWORK REPRESENTATION AND INFERENCE ALGORITHMS

This section describes the diagnostics enabling technology that we use. The diagnostic inference algorithms are based on causal networks and are enhanced using Ordered Binary Decision Diagrams (OBDDs). The causal network approach, given a system model, provides consistency and completeness guarantees for diagnostic inference [2], in addition to computational guarantees in terms of the underlying model [15,16]. The OBDD technology provides a means for further improving the inference space and time requirements, although with no firm guarantees. Together, these two technologies allow users to build complex models without worrying about scale-up issues, as has been the case in the past. Instead, the computational requirements can be assessed *a priori* from the model parameters, and optimizations introduced, such as changes to models, when necessary.

A. Causal Network Inference System

A causal network is a graph-based representation that can be used for simulating both normal and abnormal system behaviors, as well as diagnosing failures of a system under abnormal operation.

CNETS is a system for representing and reasoning with causal networks. CNETS allows the user to create a model by specifying the cause/effect relationships for the system. Once the causal model is built, a user will typically engage in the following type of scenario with CNETS. First, the user asserts available evidence about the modeled system. Second, the user makes queries about the behavior of the system in light of the available evidence and the system's model (causal network). These queries depend on the applications at hand, but they are typically simulation, prediction and/or diagnostic queries. Following are example queries that, among others, CNETS provides answers to:

- Shuttle diagnosis (Probabilistic query):

  With respect to the main Shuttle Engine, What is the probability of Engine failure given that the He Pressure is low and Ox Inlet Pressure is nominal?

- Shuttle simulation and diagnostics (Boolean query):

  With respect to an avionics control system, given that *bus-apc-5*, *bus-control-logic-12*, *bus-control-logic-13*, and *mca-status-1* are all high, what are the possible failure states of the controller? And given any of these failure states, what functionality can we expect from the controller?

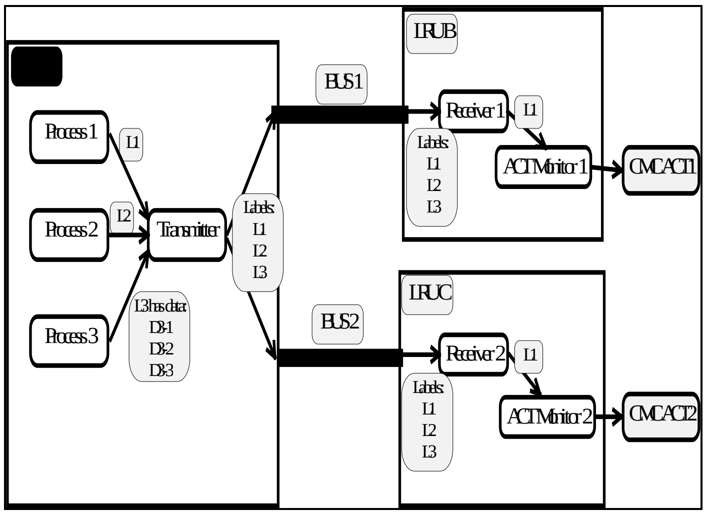


Figure 1: Simple Avionics Example

B. Causal Network Representation

A causal network model specifies a causal structure and a quantification of this structure. These two elements are now summarized.

Causal Structure

The causal structure specifies the cause/effect relationships among system variables, namely how variable X (a child) depends on its parents in the causal structure. In other words, a causal network specifies the causal relationships among a set V of variables by encoding each variable in V with a node, and encoding the causal influence of V_1 on V_2 by a directed arc from V_1 to V_2. Hence the structure of a causal network is a directed graph (N,A) of nodes N and directed arcs A.

An example of how a causal network encodes the causal relations of a system, taken from the avionics domain, is shown in Figure 1: the transmitter in LRU A causes BUS 1 and BUS 2 to carry certain labels, and BUS 1 causes Receiver 2 to receive certain labels, etc. A causal network is thus a good way of representing the flow of information in such systems, and hence reasons about the root causes of information flow; e.g. Process1 is the root cause of Label L1 being generated. Conversely, if information is not flowing as intended, the causal network can help track down the reason for the "breakdowns" of correct information flow; e.g., Process 1 could be the root cause of Label L1 being absent at LRU A and B and causing the fault reports CMC ACT1 and CMC ACT2 to be active. This is called diagnosing the faults in the system.

Causal Network Quantification

The quantification identifies the details of the causal structure, namely the values that a child variable can take on given the values of its parent variables. The chosen method of quantification depends on the application domain, the reasoning task, and the available information about the system being modeled. Figure 2 provides three different quantifications of the same causal structure, which represents a digital circuit. The given quantifications are:

- **Probabilistic quantification**:

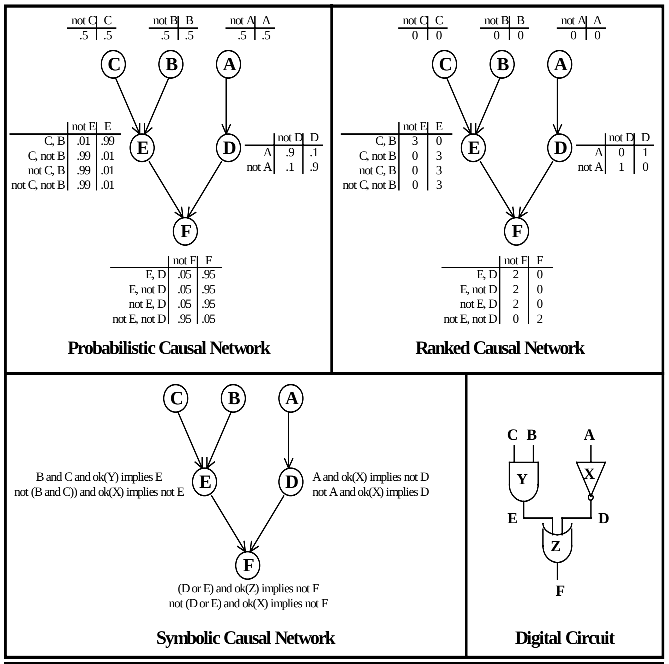
  For each state of a node and for each state of its parents, provide the conditional probability of the node given its parents. For example, in Figure 2, for each state S_F of node F, and for each state $S_{E,D}$ of its parents E and D, one must provide the probability of S_F given $S_{E,D}$.

- **Order-of-Magnitude Probabilistic (OMP) quantification:**

  For each state of a node and for each state of its parents, provide the conditional OMP of the node given its parents. An OMP (also known as a ranking) is a positive integer and represents a degree of disbelief. OMPs are typically used when exact probabilities are not available or are judged to be too detailed by domain experts.

- **Symbolic quantification**:

For each node, provide a set of logical statements (using a multivalued propositional logic) relating the node to its parents. As described below, a term of the form OK(X) denotes the mode of component X, i.e., the what operating mode X is in, such as normal or faulty.



	not C	C
	.5	.5

	not B	B
	.5	.5

	not A	A
	.5	.5

	not E	E
C, B	.01	.99
C, not B	.99	.01
not C, B	.99	.01
not C, not B	.99	.01

	not D	D
A	.9	.1
not A	.1	.9

	not F	F
E, D	.05	.95
E, not D	.05	.95
not E, D	.05	.95
not E, not D	.95	.05

	not C	C
	0	0

	not B	B
	0	0

	not A	A
	0	0

	not E	E
C, B	3	0
C, not B	0	3
not C, B	0	3
not C, not B	0	3

	not D	D
A	0	1
not A	1	0

	not F	F
E, D	2	0
E, not D	2	0
not E, D	2	0
not E, not D	0	2

B and C and ok(Y) implies E
not (B and C)) and ok(X) implies not E
A and ok(X) implies not D
not A and ok(X) implies D
(D or E) and ok(Z) implies not F
not (D or E) and ok(X) implies not F

Figure 2: Quantification of Causal Networks

These quantifications lead to probabilistic, OMP, and symbolic (or Boolean) causal networks, respectively (see Figure 2).

CNETS Specification Language

To allow a causal network to model a system such as that shown in Figure 2, we need to specify:

1. the variables in the model, which represent the components, e.g., Process 1, Receiver 1, Transmitter, Bus 1, etc.;
2. the assumables, which are the variables describing the operating characteristics of the components;
3. the quantification, e.g., for a symbolic network the equations relating the variables and assumables;
4. the evidence variables, which are the variables that can be observed;
5. the weights for the assumables, which specify the likelihood or ranking of the assumable fault modes.

Some further clarification about the component modes: they characterize the ways the component works under an exhaustive set of scenarios. Thus, we might say that Bus 1, if OK, transmits labels correctly, but if broken transmits no data. In Boolean logic, we might write this as:

IF (input-label = X) AND (bus1-mode = OK) THEN (output-label = X)
IF (input-label = X) AND (bus1-mode = broken) THEN (output-label = no-data).

Causal Network Inference

CNETS possesses a number of inference algorithms that can be used to process such queries. The fundamental class of algorithms that CNETS uses to transform the input causal network into a form for facilitating the particular type of query-processing. The main algorithm used at present is a modified version of the clique-tree algorithm [10]. If the queries are probabilistic and the user has access to CNETS on a machine with a powerful CPU, then an algorithm like the clique-tree algorithm can be used directly. CNETS also supports a Boolean form of this algorithm for answering Boolean or symbolic queries.

D. CNETS Compiler

CNETS has a compiler that allows the user to generate a run-time version of the system model by using a compiler in conjunction with an algorithm like [10]. This way diagnostic queries can be performed on embedded platforms with limited processing power. This compiler is described in detail in [9].

Using this novel compilation technique [9], CNETS is able to compile a causal network into a Boolean expression, called a Query-DAG, or Q-DAG, that contains all possible diagnoses for the system, given inputs for the observable variables of the system. The Q-DAG allows the use of a trivial evaluator in the embedded system that greatly reduces the processor and memory requirements, as compared to the original causal network model and CNETS software.

E. Complexity Guarantees

One of the key features of the causal network approach is that, for diagnostic inference, it can provide complexity guarantees in terms of the model parameters. The structure of the network is the most critical aspect that governs the complexity of inference, and this is why causal networks are one of several classes of techniques termed "structure-based" approaches. Hence, in the causal graph the integer n_P, the maximum of the parent nodes of any node, is one of the most significant parameters. A more precise parameter, called a clique table, is a function of n_P: when the causal graph is transformed into a tree of cliques (where the maximum clique size is closely related to n_P), the clique table is the product of the variable domain sizes of the clique variables.

Because all significant complexity parameters are ultimately derivable from the system structure, one can manipulate the structure to obtain acceptable inference complexity without altering the diagnostic coverage. Approaches that are not structure-based do not have this capability to alter inference complexity without altering the diagnostic coverage.

Beyond the complexity guarantees, the causal network approach has been optimized in a number of ways. This approach uses a focusing mechanism to focus inference on only the most likely diagnoses [2], and it makes use of observations to decompose the system (and hence reduce overall inference complexity) [16].

Ordered Binary Decision Diagrams

Causal networks offer one technique for representing a system for diagnostic purposes. An alternative approach uses *fault propagation graphs* (FPGs), that model the system in terms of interconnected components, their failure modes, and propagation links that describe how failures interact. FPGs can have propagation delays associated with their edges, thus provide a way for modeling dynamic behavior. Efficient graph-algorithms can be used to quickly calculate a "diagnosis": an explanation for an evolving fault scenario[11]. These algorithms are polynomial in the number of nodes of the graph. However, if arbitrary logical connectives are allowed among the fault modes, (AND, OR, NOT), the algorithms become exponential leading to combinatorial explosion. However, a different approach offers an engineering solution to the problem.

Diagnosis and fault analysis tasks with discrete models can be formulated in terms of operations over finite domains. Combinatorial explosion is the result of the exponential increase in the number of discrete elements (states, hypotheses, etc.) during operations, which sooner or later makes the individual access to the elements unfeasible. By introducing a binary encoding, the individual elements, sets of elements, and relations among them can be expressed as Boolean functions. For example, the 2^{100} states of a finite state automaton can be encoded with binary variables $\{s(1),\ldots,s(100)\}$ forming a binary state vector $\underline{\boldsymbol{s}}$. The Boolean functions

$f_1[s(1),\ldots,s(100)]=s(1)'\wedge s(23)\wedge s(99)$ and

$f_2[s(1),\ldots,s(100)]=s(1)\wedge s(22)\wedge s(89)$

represent two subsets, S1 and S2, of the 2^{100} states including 297 elements each. The set $S3=S1\cup S2$ can be derived symbolically as the disjunction of the two Boolean functions:

$$f_3[s(1),\ldots,s(100)]= f_1[s(1),\ldots,s(100)] \vee$$
$$f_2[s(1),\ldots,s(100)]= s(23)\wedge s(99)\vee s(22)\wedge s(89)$$

without the need to enumerate and compare the individual elements - which would be a formidable task otherwise. In general, using Boolean function representations, we can express operations and algorithms in diagnosis and safety analysis in symbolic form, by means of symbolic Boolean function manipulations.

OBDDs provide a symbolic representation for Boolean functions in the form of directed acyclic graphs, and are a restricted, canonical form version of Binary Decision Diagrams (BDD) [3]. Bryant [3] described a set of algorithms that implement operations on Boolean functions as graph algorithms on OBDDs. Taking advantage of the efficient symbolic manipulations, researchers have solved a wide range of problems in hardware verification, testing, real-time systems, and mathematical logic using OBDDs that would have been otherwise impossible due to combinatorial explosion. Symbolic model checking is extensively used in hardware design (see, e.g., [11]), and has shown to be efficient in state space sizes 10^{120} and beyond.

Discrete Event System and Relational Models for Diagnosis

A broad category of systems, such as digital hardware, switching, distribution, and communication networks, etc., are naturally modeled as DES. Beyond this, the behavior of continuous systems can also be approximated with DES models.

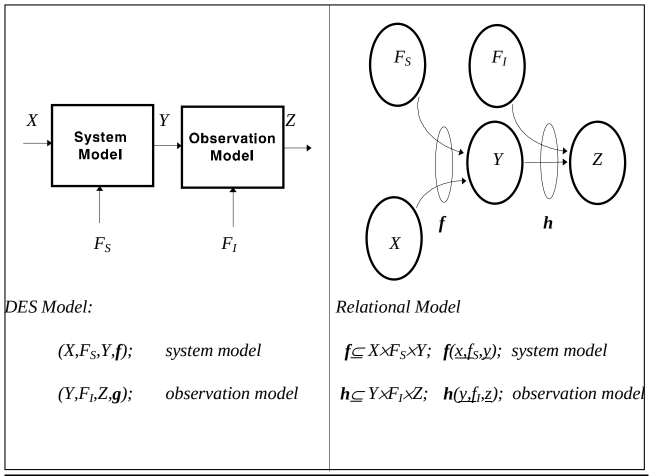

X
System Model
Y
Observation Model
Z
F_S
F_I
F_S
F_I
Y
Z
X
f
h
DES Model:
$(X,F_S,Y,\boldsymbol{f})$; system model
$(Y,F_I,Z,\boldsymbol{g})$; observation model
Relational Model
$\boldsymbol{f}\subseteq X\times F_S\times Y$; $\boldsymbol{f}(\underline{x},\underline{f_S},\underline{y})$; system model
$\boldsymbol{h}\subseteq Y\times F_I\times Z$; $\boldsymbol{h}(\underline{y},\underline{f_I},\underline{z})$; observation model

Figure 3: DES and relational models of static systems

The DES model of a static system (system without memory) is shown on the left side of Figure 1. Since our purpose with modeling is sensor-based diagnosis, the model is divided into a *System model* and an Observation *model*. The system model represents a mapping between the elements of the input set X, fault set F_S, and the elements of the output set Y: $\boldsymbol{f}$: $X\times F_S \rightarrow Y$. In this approach, the component faults are considered as additional inputs to the system. It is also possible to model the abnormal (out of range) inputs as elements of the X input set, creating a 'normal' and 'faulty' partition in X. The observation model describes a mapping between the Y output set of the system, and the actually observed outputs, Z; $\boldsymbol{h}$: $Y\times F_I \rightarrow Z$. The set F_I collects the observation faults (or instrumentation faults) that can potentially corrupt the observations. We assume that both $\boldsymbol{f}$ and $\boldsymbol{h}$ can represent many-to-many mapping, i.e. they are not necessarily functions. This allows non-deterministic modeling, which is particularly important in large-scale systems. Non-deterministic constructs allow the expression of uncertainties in the outcome of inputs due to noises or non-modeled behaviors.

The right side of Figure 1 shows the equivalent *Relational Model* of a static system. In the relational model, the $\boldsymbol{f}$ and $\boldsymbol{h}$ mappings are considered to be the $\boldsymbol{f} \subseteq X \times F_S \times Y$ and $\boldsymbol{h} \subseteq Y \times F_S \times Z$ relations. The significance of the relational representation is that it directly shows that the models can be re-written as Boolean functions by introducing some binary encoding for the sets $X \times F_S \times Y$ and $Y \times F_I \times Z$. The Boolean functions $\boldsymbol{f}(\underline{x},\underline{f_S},\underline{y})$ and $\boldsymbol{h}(\underline{y},\underline{f_I},\underline{z})$ evaluate to *true* for those elements of $X \times F_S \times Y$ and $Y \times F_I \times Z$ (encoded by the Boolean vectors $(\underline{x},\underline{f_S},\underline{y})$ and $(\underline{y},\underline{f_I},\underline{z})$), which are related by the $\boldsymbol{f}$ and $\boldsymbol{h}$ relations.

The DES model and relational model of a dynamic system are more complicated, as shown in Figure 4.

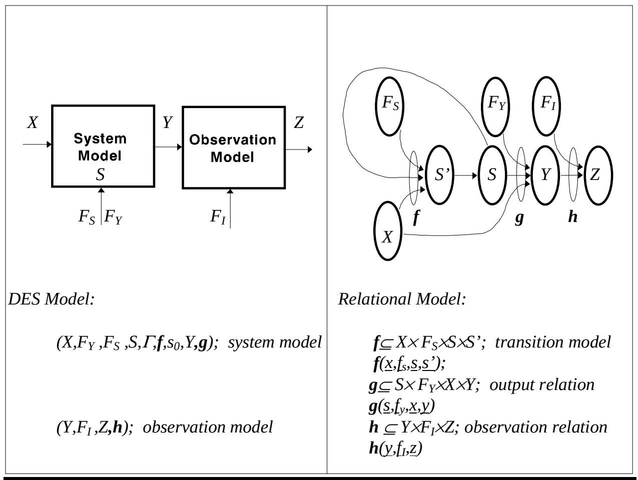

X
Y
Z
System Model
S
Observation Model
F_S
F_Y
F_I
F_S
F_Y
F_I
S'
S
Y
Z
X
f
g
h
DES Model:
(X,F_Y ,F_S ,S,Γ,f,s_0,Y,g); system model
(Y,F_I ,Z,h); observation model
Relational Model:
f⊆X×F_S×S×S'; transition model
f(x,f_s,s,s');
g⊆S×F_Y×X×Y; output relation
g(s,f_y,x,y)
h ⊆Y×F_I×Z; observation relation
h(y,f_I,z)

Figure 4: DES and relation models for dynamic systems

In dynamic systems, the DES model is the $(X,F_Y ,F_S ,S,\Gamma ,\boldsymbol{f},s_0,Y,\boldsymbol{g})$ finite state automata (see e.g. [12]), where:

X is the input event set,

F_S ,F_Y are the sets of transition faults and output faults, both considered to be input events,

S is the state set,

$\Gamma(s)$ is a set of feasible or enabled events, defined for all $s \in S$ with $\Gamma(s) \in X$,

$\boldsymbol{f}$ is a state transition function, $\boldsymbol{f}: X \times F_S \times S \rightarrow S'$, defined only for $x \in \Gamma(s)$ when the state is s,

s_0 is the initial state,

Y is the output set, and

$\boldsymbol{g}$ is an output function, $\boldsymbol{g}: X \times F_Y \times S \rightarrow Y$, defined only for $x \in \Gamma(s)$ when the state is s.

In order to model partial observations of the state trajectory independently from the outputs of the dynamic system, we use again the $\boldsymbol{h}: Y \times F_I \rightarrow Z$ observation model describing the mapping between the Y outputs, F_I instrumentation faults, and the Z observations. The finite state automaton formalism also allows the representation of non-deterministic state machines, which is an important requirement for modeling large-scale systems.

The right side of Figure 2 shows the equivalent relational models. Similarly to the static system models, the $\boldsymbol{f}(\underline{x},\underline{f_s},\underline{s'},\underline{s})$, $\boldsymbol{g}(\underline{s},\underline{f_y},\underline{x},\underline{y})$ and $\boldsymbol{h}(\underline{y},\underline{f_I},\underline{z})$ functions are the Boolean function representations of the relations over the binary encoded Boolean vectors $\underline{x},\underline{f_s},\underline{s'},\underline{s},\underline{f_y},\underline{y},\underline{f_I}$, and $\underline{z}$.

Although it is not the purpose of this discussion, it is worthwhile to note that DES (or relational) models preserve composability and can be constructed in a modular fashion using either component oriented modeling approach [13] or process-oriented modeling approach [14].

DIAGNOSTIC REASONING USING OBDDS

The application of OBDDs for diagnostic reasoning includes the following steps:

1. Mapping the DES or relational models into OBDD-s: This step can be completed automatically. In the framework of the Multigraph Architecture (MGA), the discrete behavioral models used for diagnosis or safety analysis are usually domain specific [4]. The domain specific models can be translated into an OBDD representation using model interpreters. An example for converting relay logic diagrams into OBDD representations is described in [13,14]
2. Diagnosability And Safety Analysis: Diagnosability and safety analyses are accomplished symbolically, using the OBDD representations. Diagnosability and safety criteria are expressed in the form of logic relationships on the discrete state trajectories generated by the models, and these relationships are checked using OBDD algorithms.
3. Diagnosis: In a model-integrated framework, MGA [23], the diagnostic software is built in two steps. First, a generic run-time support is created. The run-time support includes a diagnostic engine implemented with OBDD algorithms. Second, the software synthesis component of the MGA (one particular form of model interpreters): (a) configures the run-time system using the MGA computational model, and (b) synthesizes the OBDD data structures for the models. The core components of the diagnostic reasoning are the algorithms that compute the sets using the relational models.

Diagnostic reasoning in static systems

Although it seems to be restrictive, static system models are widely used in engineering practice. Fault trees, AND-OR graphs, most of the rule-based models can be considered as some form of the static models. Using the relational model formulation described above, the following calculations can be performed using OBDD algorithms.

a) **Observed output calculation:** Given the set of input X, and the sets of faults F_S and F_I, the set of outputs Y and the set of observations Z can be calculated by the following formulae:

$$Y = \boldsymbol{f}(X,F_S)=\{y \mid \exists x, f_S[(x \in X) \wedge (f_S \in F_S) \wedge (x, f_S,y) \in \boldsymbol{f}]\}; \quad (1)$$
$$Z=\boldsymbol{h}(Y,F_I)=\{z \mid \exists y,f_I[(y \in Y) \wedge (f_I \in F_I) \wedge (y, f_I,z) \in \boldsymbol{h}]\};$$

The required variable quantification and logic operations are executed symbolically. The resulting $\boldsymbol{f}(X,F_S)$ and $\boldsymbol{h}(Y,F_I)$

mappings propagate the elements of the input sets 'forward' in the relational model.

b) **Diagnosis:** Solution of the diagnosis problem requires the calculation of the hypothesis set D, defined on $X{\times}F_S{\times}F_I$ given a set of observations Z:

$$\boldsymbol{d}(Z)=\{x,f_S,,f_I|\exists y,z[(z\in Z)\wedge(y,f_I,z)\in h\wedge(x,f_S,y)\in\boldsymbol{f}]\} \qquad (2)$$

The diagnostic mapping is derived as a combination of the functional composition of the relations $\boldsymbol{f}$ and $\boldsymbol{h}$, and variable quantification. The $\boldsymbol{d}(Z)$ mapping propagates the Z observations 'backwards' to obtain the set of admissible $d\in X{\times}F_S{\times}F_I$ elements forming together the diagnosis. The result of the diagnosis, the D hypothesis set, includes all of those $d\in X{\times}F_S{\times}F_I$ hypotheses that are consistent with the Z observations. Those elements for which $f_S=\{0\}$ and $f_I=\{0\}$, the corresponding x input values represent inputs for fault free operations. *It is interesting to note that the symbolic computation derives in one step the symbolic representation (i.e. the OBDD) of the full D(Z) hypothesis set, including all of the multiple fault combinations.*

c) **Safety analysis**: Safety analysis requires testing models against selected safety criteria. Here we demonstrate the use of symbolic model checking in one particular problem, to test distinguishability of faults. A system and its observation model provide *single-fault distinguishability*, if all possible observations are unique to the single faults. Let $i_{fs}\in X{\times}F_S{\times}f_I$ be inputs to the system with a single fault, i.e., each i_{fs} includes exactly one f_S or f_I faults. The condition for single fault distinguishability can be expressed symbolically using:

$$\boldsymbol{d}^{\circ}(\boldsymbol{h}\circ\boldsymbol{f})(i_{fs})=i_{fs}\,; \quad \forall\, i_{fs}\in X{\times}F_S{\times}F_I \qquad (3)$$

That is, the diagnosis relation $\boldsymbol{d}$ is the inverse of the composition of the $\boldsymbol{h}\circ\boldsymbol{f}$ relations for all single fault inputs. Similar symbolic expressions can be derived for multiple fault distinguishability, fault masking, fault detectability and other safety characteristics of the models.

Diagnostic Reasoning In Dynamic Systems

The primary difficulty in dynamic systems is that often diagnoses are computed based on only partial observations of the system trajectory. Particularly in the case of non-deterministic models, the diagnostic reasoning has acute scaling problems. The symbolic form of the algorithms has a similar form to that of static systems.

a) **Observed output calculation**: Given the set of input X and the sets of faults F_S, F_Y and F_I, the sets of next states S', outputs Y and observations Z can be calculated by the following expression:

$$S=\boldsymbol{f}(X,F_S,S)=\{s'|\exists x,f_S,s[(x\in X)\wedge(f_S\in F_S)\wedge(s\in S)\wedge(x,f_S,s,s')\in\boldsymbol{f}]\};$$
$$Y=\boldsymbol{g}(X,F_Y,S)=\{y|\exists x,f_Y,s[(x\in X)\wedge(f_Y\in F_Y)\wedge(s\in S)\wedge(x,f_Y,s,y)\in\boldsymbol{g}]\};$$
$$Z=\boldsymbol{h}(Y,F_I)=\{z|\exists y,f_I[(y\in Y)\wedge(f_I\in F_I)\wedge(y,f_I,z)\in\boldsymbol{h}]\}; \qquad (4)$$

The symbolic expression above calculates a one-step propagation forward in the state automata. The result is a new set of possible states S', and the related Y outputs and Z observations. The set of reachable states can be found by computing the transitive closure of $\boldsymbol{f}$ using fixed-point calculation, i.e. to find an S for which $\boldsymbol{f}(X,F_S,S)=S$. This is particularly important in safety analysis, where safety requirements frequently impose constraints on the reachability set of the state automata.

b) **Diagnosis:** There are several ways to perform diagnosis in dynamic systems. In off-line diagnosis, observations are collected and analyzed independently from the operation of the system. In on-line diagnosis, the diagnostic system runs parallel with the system, and refines the hypothesis set as new observations are collected. As an example, we describe an on-line diagnosis method using symbolic expressions. The on-line diagnostic system re-calculates the hypothesis set $D\subset X{\times}F_S{\times}F_Y{\times}F_I{\times}S$ whenever a new observation event(s) arrives:

$$D_{j+1}=\boldsymbol{d}(D_j,Z_{j+1})=$$
$$\{x,f_S,,f_Y,f_I,s'|\exists s,y,z[(z\in Z_{j+1})\wedge(y,f_I,z)\in\boldsymbol{h}\wedge(f_I\in F_{I,j})\wedge$$
$$\wedge(\underline{s},\underline{f_y},\underline{x},\underline{y})\in\boldsymbol{g}\wedge(x\in X_j)\wedge(f_Y\in F_{Y,j})\wedge(x,f_S,,s,s')\in\boldsymbol{f}\wedge(f_S\in F_{S,j})]\} \quad (5)$$

The $(x\in X_j)$, $(f_S\in F_{S,j})$, $(f_Y\in F_{Y,j})$, and $(f_I\in F_{I,})$ conditions assume that during the observation the faults (including possibly faulty x inputs) are persistent. If this assumption cannot be made, the conditions must be eliminated from the reasoning. Receiving newer and newer observations, the diagnostic algorithm will converge to a hypothesis set that includes all possible explanations for the observed trajectory. These explanations extend to the multiple fault hypotheses as well.

Criteria for fault distinguishability can be derived the same way as it was described for static systems.

It is important to mention that the diagnostic algorithms of *(2)* and *(5)* (and all of the other expressions above) are computed symbolically. Symbolic computation here means that all of the sets and relations are represented as OBDDs and the logic and quantification operators are executed by manipulating the OBDDs by means of a small set of efficient algorithms [3].

INTEGRATING CNETS AND OBBD-BASED DIAGNOSTIC REASONING

A causal network is a graph-based representation that can be used for simulating both normal and abnormal system behaviors for discrete event systems. The expected discrete behavior of a given system is captured and modeled using causal networks. Causal networks provide a generalized modeling paradigm that can serve to populate other modeling paradigms for many different types of analyses.

For example, the failure propagation topology of a system may be extracted from the causal network model. This topology can be translated into a relational model and ultimately into OBDDs. Thus, starting from a causal network model of a system, all the advantages of the OBDD-based analyses can be used, including static and dynamic analyses of failure propagation graphs.

The complexity guarantees of the CNETS diagnostic inference engine enable one to predict throughput requirements. This is a crucial ingredient in the design process. Moreover, OBDDs may be used to even further reduce the complexity of the diagnostic inference engine.

The upper bound complexity guarantees of the CNETS diagnostic inference engine coupled with the highly scaleable properties of OBDDs make the integration of these two technologies eminently practical.

CONCLUSIONS AND FUTURE WORK

Future work will focus on reapplying the concepts outlined in this paper into an integrated modeling paradigm that will support translations to other modeling paradigms and thus bridge the previously mentioned analyses and others. Difficult issues in the design and implementation of diagnostic inference engines for complex systems have been explicitly addressed. In particular, scalability, predictable complexity, dynamic and static system modeling, exhaustive symbolic analyses, and a quasi-domain-independent modeling paradigm (causal networks) have all been addressed. Finally, all the technologies discussed herein have been proven through their implementation on real life complex systems.

REFERENCES

[1] Janos Sztipanovits, Amit Misra: "Diagnosis Of Discrete Event Systems Using Ordered Binary Decision Diagrams", Proc. Of The 7th International Workshop On Principles Of Diagnosis (Dx96), Pp. 232-238 Val Morin, Quebec, Canada, October 13-16, 1996.

[2] A. Darwiche: "New Advances In Structured-Based Diagnosis: A Method For Compiling Devices", Proc. Of The 8th International Workshop On Principles Of Diagnosis (Dx97).

[3] R. E. Bryant, Binary Decision Diagrams And Beyond: Enabling Technologies for Formal Verification. Embedded Tutorial At International Conference On Computer-Aided Design November, 1995.

[4] Sztipanovits, J., Karsai, G., Biegl, C., Bapty, T., Ledeczi, A., Misra, A., "MULTIGRAPH: An Architecture for Model-Integrated Computing," Proc. of the ICECCS'95, pp. 361-368, Ft. Lauderdale, Florida, Nov. 6-10, 1995.

[5] A Unifying Theoretical Background For Some BDD-Based Data-Structures By Meinel-C Slobodova-A, Form-Meth-S V11 (3) : Pp223-237 (1997 Oct)

[6] Misra, A., Sztipanovits, J., Carnes, R.: "Robust Diagnostic System: Structural Redundancy Approach," in Proc. of the SPIE's International Symposium on Knowledge-Based Artificial Intelligence Systems in Aerospace Systems in Aerospace and Industry, Orlando, FL, April 5-6, 1994.

[7] Sztipanovits, J., Wilkes, D., Karsai, G., Biegl, C., Lynd, L: "The Multigraph and Structural Adaptivity," IEEE Transactions on Signal Processing, Vol. 41, No. 8., pp. 2695-2716, 1993.

[8] Rudell, "Dynamic Variable Ordering for Ordered BDDs", *Proceedings of the International Conference on Computer-Aided Design*, 1993.

[9] Adnan Darwiche and Gregory Provan, "Query-DAGs: A Practical Paradigm for Implementing Belief-Network Inference," *Journal of AI Research*, 1996.

[10] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen, "Bayesian Updating in Recursive Graphical Models by Local Computation," *Computational Statistics Quarterly*, 4:269--282, 1990.

[11] Burch, J. R., Clarke, E.M., Long, D. E., "Symbolic Model Checking for Sequential Circuit Verification," Technical Report, CMU-CS-93-211, Carnegie Mellon University, July 15, 1993.

[12] Cassandras, C. G., *Discrete Event Systems*, IRWIN Inc. 1993.

[13] Sampath, M. et al., "Failure Diagnosis Using Discrete-Event Models," *IEEE Transactions on Control Systems Technology.* Vol. 4, No. 2, pp. 105-124, March, 1996.

[14] Sztipanovits, J., Carnes, R., Misra, A., "Finite-State Temporal Automata Modeling for Fault Diagnosis," *Proc. Of the 9th AIAA Conference on Computing in Aerospace*, San Diego, CA, October, 1993.

[15] A. Darwiche: " Compiling Devices: A Structured-Based Approach", Proc. Of Conference On Knowledge Representation (KR98).

[16] Adnan Darwiche and Gregory Provan, "The Effect of Observations on the Complexity of Model-based Diagnosis," *Proc. AAAI, pp. 91-94, 1997.*