Probabilistic Model-Based Diagnosis: An Electrical Power System Case Study

Ole J. Mengshoel, *Member, IEEE*, Mark Chavira, Keith Cascio, Scott Poll, Adnan Darwiche, and Serdar Uckun, *Member, IEEE*

***Abstract*—We present in this paper a case study of the probabilistic approach to model-based diagnosis. Here, the diagnosed system is a real-world electrical power system (EPS), i.e., the Advanced Diagnostic and Prognostic Testbed (ADAPT) located at the NASA Ames Research Center. Our probabilistic approach is formally well founded and based on Bayesian networks (BNs) and arithmetic circuits (ACs). We pay special attention to meeting two of the main challenges often associated with real-world application of model-based diagnosis technologies: model development and real-time reasoning. To address the challenge of model development, we develop a systematic approach to representing EPSs as BNs, supported by an easy-to-use specification language. To address the real-time reasoning challenge, we compile BNs into ACs. AC evaluation (ACE) supports real-time diagnosis by being predictable, fast, and exact. In experiments with the ADAPT BN, which contains 503 discrete nodes and 579 edges and produces accurate results, the time taken to compute the most probable explanation using ACs has a mean of 0.2625 ms and a standard deviation of 0.2028 ms. In comparative experiments, we found that, while the variable elimination and join tree propagation algorithms also perform very well in the ADAPT setting, ACE was an order of magnitude or more faster.**

***Index Terms*—Aerospace, arithmetic circuits (ACs), Bayesian networks (BNs), domain modeling, electrical power systems (EPSs), knowledge engineering, model-based diagnosis, real-time systems, uncertainty.**

I. Introduction

IN THIS paper, we apply probabilistic model-based diagnosis techniques to a real-world electrical power system (EPS), i.e., the Advanced Diagnostic and Prognostic Testbed (ADAPT) [1]. In this application, a Bayesian network (BN) model [2] of the ADAPT EPS plays a central role. The ADAPT BN explicitly represents the health of the sensors and subsystem components and is autogenerated from a high-level system model of the ADAPT EPS. This BN is then compiled, offline, into an arithmetic circuit (AC) that is then evaluated online. We believe that this ADAPT case study clearly demonstrates how ACs offer a scalable inference technique with potential for real-time evaluation in aircraft and spacecraft.

In several ways, this work is different from previous diagnosis efforts that utilize BNs, including EPS diagnosis [3], [4]. A first contribution is our expression of EPS components and structure, using a novel high-level language, coupled with autogeneration of BNs from models expressed in this language. This approach supports the iterative development of probabilistic diagnostic models for large EPSs, including diagnostic system models that would be extremely tedious to hand-construct, even for BN experts. The benefit of this approach to developers less familiar with BNs appears to be even greater.

As a second contribution, we highlight our offline compilation of BNs to ACs [5], [6], which are then used for online diagnosis. It is important to achieve real-time performance in many system health monitoring applications, e.g., in aerospace [7]–[10], and at the same time, the problem of diagnosis in BNs is NP-hard or worse [11]–[13]. An AC, which typically is large but has a simple and exact semantics, supports real-time diagnosis by providing fast and predictable inference times. These three benefits—fast, predictable, but exact—are very important, given the real-time requirements of aircraft and spacecraft avionics [8]–[10]. In experiments, we have successfully shown here that AC evaluation (ACE) provides high-quality diagnostic results on ADAPT scenarios. In addition, we have shown that performance is substantially better than alternative probabilistic inference algorithms, specifically variable elimination (VE) and join tree propagation (JTP).

EPSs are of crucial importance in aerospace, as well as in numerous other application [1], [14]. Our results in this paper provide an argument for the feasibility of probabilistic model-based diagnosis of complex EPSs. One of our main contributions is the integration of different techniques, both existing and novel, in order to address a real-world problem, thereby obtaining an approach that scales up to handle real-world challenges in probabilistic model-based diagnosis. Building on the approach discussed here, we have developed BNs that achieved the best overall scores in the Industrial Track of the 2009 DX Challenge Competition [15]. In addition, we have demonstrated scalability for BNs representing 24 distinct EPSs, where the largest BN had 1018 nodes and 1194 edges [16].

The rest of this paper is structured as follows. We discuss EPSs in aerospace, the ADAPT testbed, and diagnostic challenges in Section II. We then briefly introduce fundamentals of

Manuscript received April 26, 2008. Date of publication July 19, 2010; date of current version August 18, 2010. This paper was recommended by Guest Editor G. Provan.

O. J. Mengshoel is with Carnegie Mellon University, Silicon Valley, NASA Research Park, Moffett Field, CA 94305 USA, and also with the Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035 USA (e-mail: Ole.Mengshoel@sv.cmu.edu).

M. Chavira is with Google, Santa Monica, CA 90401 USA (e-mail: chavira@cs.ucla.edu).

K. Cascio and A. Darwiche are with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA (e-mail: keith@cs.ucla.edu; darwiche@cs.ucla.edu).

S. Poll is with the Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035 USA (e-mail: Scott.Poll@nasa.gov).

S. Uckun is with the Embedded Reasoning Area, Palo Alto Research Center, Palo Alto, CA 94304 USA (e-mail: Serdar.Uckun@parc.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2010.2052037

1083-4427/$26.00 © 2010 IEEE

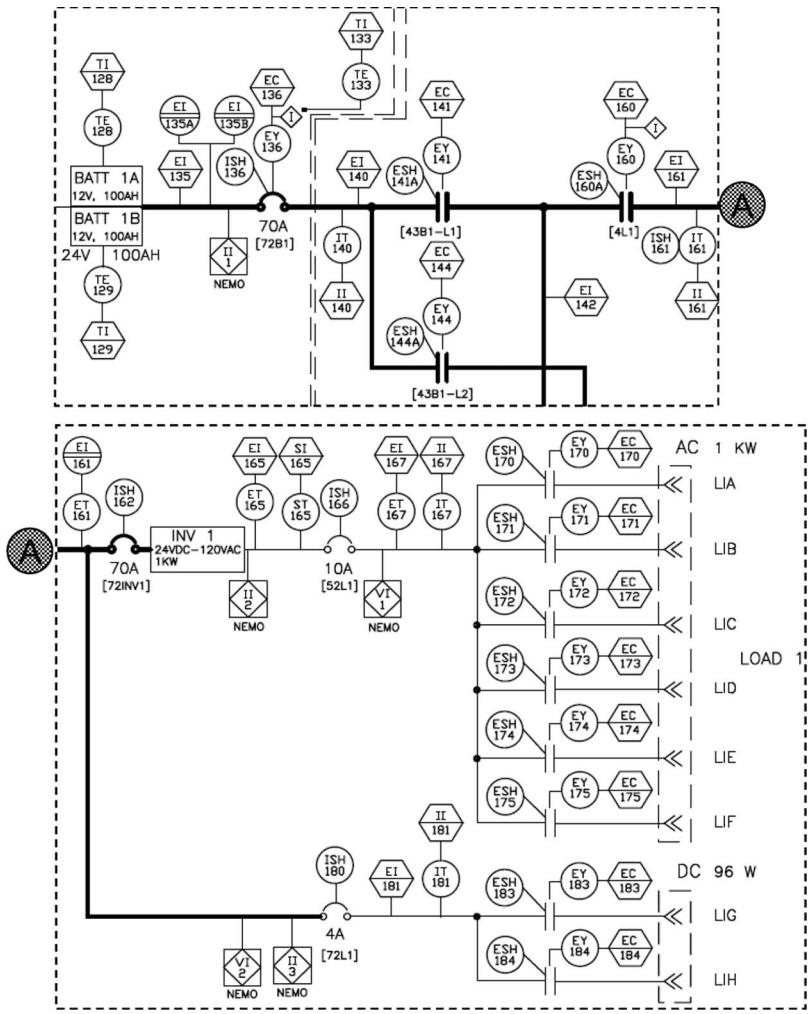


Fig. 1. Schematic of the ADAPT testbed, showing one of three power storage parts (for Battery 1, top) and one of two load banks (Load bank 1, bottom). Detailed information about loads is given in Table II.

BNs and ACs in Section III. Using ADAPT as a case study, we discuss the high-level specification language (Section IV), BN modeling and autogeneration (Section V and Section VI, respectively), and compilation to ACs (Section VII). Finally, we report on experimental results in Section VIII, both on real-world and synthetic data, before concluding in Section IX.

II. EPSs

We consider the importance of EPSs in aerospace, describe an EPS testbed that is the subject of this case study, and discuss diagnostic challenges associated with EPSs.

A. Challenges in Aerospace and at NASA

The essential role that EPSs play in aerospace vehicles is well known [1], [14]. The EPS may be thought of as the circulatory system of an aerospace vehicle. In the human body, the circulatory system delivers oxygen and removes carbon dioxide. Similarly, the EPS delivers energy to subsystems in order to power required vehicle functions, such as life support, propulsion, communications, guidance, navigation, and control. Loss of electrical power to these and similar subsystems can have severe repercussions for the vehicle, personnel, or mission.

Unfortunately, EPSs have been implicated in several aerospace vehicle incidents, accidents, and mishaps. In one accident, the left power conversion and distribution unit (PCDU) on a Boeing 717 failed, resulting in the loss of the left AC and DC buses. The most likely cause was determined to be the failure of a transient suppression diode, which allowed AC current to contaminate the DC circuits of the PCDU. In another incident involving the PCDU of a Boeing 717, a tantalum capacitor and a permanent-magnet generator input transformer failed, resulting in smoke in the cabin and an emergency landing and evacuation (NTSB report ATL04IA085). The Electric propulsion Space EXperiment (ESEX) mission, which was launched and operated in early 1999, prematurely ended when the spacecraft experienced a catastrophic battery failure. The failure was most likely the result of electrolyte leakage, which caused a short circuit to the battery case, resulting in a breach of the battery case, entry of superheated gas into the flight unit, and eventual venting into space [17]. On January 14, 2005, an Intelsat-operated communications satellite suffered a total loss after a sudden and unexpected EPS anomaly. The failure of

TABLE I
DIFFERENT EPS PARTS, ALONG WITH THEIR MODES AND THE CORRESPONDING STATES OF THE HEALTH NODE FOR THE PART

Part	Prefix	Mode (Healthy/Faulty)	State
Battery	BATT	Healthy Voltage failure or drain	*healthy* *stuckDisabled*
Circuit breaker	ISH	Healthy Stuck or failed open Stuck or failed closed	*healthy* *stuckOpen* *stuckClosed*
Inverter	INV	Healthy Switched off	*healthy* *stuckOpen*
Relay	EY	Healthy Stuck or failed open Stuck or failed closed	*healthy* *stuckOpen* *stuckClosed*
Voltage sensor	EI	Healthy Reading stuck low Reading stuck high	*healthy* *readVoltageLo* *readVoltageHi*
Current sensor	IT	Healthy Reading stuck low Reading stuck high	*healthy* *readCurrentLo* *readCurrentHi*
Position sensor	ISH	Healthy Reading stuck open Reading stuck closed	*healthy* *stuckOpen* *stuckClosed*

Intelsat 804's high-voltage power system was likely the result of a high-current event in the battery circuitry triggered by an electrostatic discharge (see http://sat-nd.com/failures/index.html/). A battery failure also occurred on the Mars Global Surveyor, which last communicated with earth on November 2, 2006. A software error oriented the spacecraft to an angle that exposed it to too much sunlight. This caused the battery to overheat and ultimately led to the depletion of both batteries (see http://mpfwww.jpl.nasa.gov/mgs/newsroom/20070413a.html).

These are just a few examples of the faults that can arise in EPSs. Given the prevalence and importance of EPSs, it is vital to develop effective health management approaches, including diagnostic techniques, for operation in aerospace vehicles.

B. ADAPT

We now turn to ADAPT (see also http://ti.arc.nasa.gov/adapt/ and [1]). ADAPT, which has capabilities for power generation, power storage, and power distribution, is a fully operational EPS that is representative of such systems in aircraft and spacecraft. Fig. 1 presents a schematic with a representative battery and load bank from ADAPT. ADAPT is configured to achieve fault tolerance and contains three batteries and two load banks. One battery can provide power to two load banks. However, two batteries may not power the same load banks simultaneously. In Fig. 1, e.g., for Battery 1 to power Load bank 1, relay EY141 is closed. For Battery 1 to power Load bank 2, on the other hand, relay EY144 is closed. Relays EY141 and EY144 cannot be both closed at the same time.

Different types of components and sensors used in ADAPT are presented in Table I. Relays, which are commanded to close and open to control power, have prefix EY (in Fig. 1) and health modes as indicated in the table. A position sensor, which is also presented in Table I, reports on the status of a relay. As concrete examples, consider in Fig. 1 relay EY170 that controls power to load L1A; it also has a position (or touch) sensor ESH170. Our probabilistic diagnostic application works on real-world data from ADAPT. In our application, each of EY170 and ESH170 are represented by random variables, including random variables representing health state, as shown in Table I. For example, EY170's health random variable has states *{healthy, stuckOpen, stuckClosed}*. Upstream of relay EY170 is a current sensor IT167; the states of its health variable are *{healthy, readCurrentLo, readCurrentHi}*, as shown in the table. Further information on modeling of EPS components and structure is provided in Sections V and VI.

TABLE II
LOADS AND THEIR SENSORS (WHERE APPLICABLE) FOR LOAD BANK 1 OF THE ADAPT ELECTRICAL POWER SYSTEM

ID	Relay	Description	Load	Measurements (Sensors)
L1A	EY170	3 light bulbs	LGT6	Temperatures (TE500, TE501,TE502); Light sensor (LT500)
L1B	EY171	Big fan	FAN1	RPM (ST515)
L1C	EY172	Small fan	FAN3	None
L1D	EY173	1 light bulb	LGT8	None
L1E	EY174	Water pump	PMP2	Flow rate (FT525)
L1F	EY175	1 light bulb	LGT4	Temperature (TE511)
L1G	EY183	Electromech.	DC1	None
L1H	EY184	None	N/A	N/A

There are two load banks in ADAPT, and each has an AC part and a DC part. Load bank 2 is very similar to Load bank 1; the loads are just plugged into different locations. There is no ambiguity as to which "power outlet" a load is "plugged into." At this time, there are mostly AC loads in ADAPT (see Table II). Currently there are two DC loads, one for each load bank. To convert DC power from the batteries into AC power used by the AC loads, ADAPT has two inverters, one per load bank. A failed inverter breaks power transmission to the AC loads; see the *stuckOpen* failure state in Table I.

C. Diagnostic Challenges

There are several diagnostic challenges associated with EPSs including ADAPT. First, they often have a large number of distinct modes due to mode-inducing components, such as relays, circuit breakers, and loads. If an EPS has m such components and we conservatively assume two discrete states for each, there are potentially 2^m modes in the EPS. Second, while much EPS behavior is deterministic, there is both sensor noise and system state uncertainty in EPSs. Sensor noise is due to the imperfections of sensing, whereas system state uncertainty is due to failures of EPS components and sensors. Third, the mode switching behavior of EPSs often induces transients in system response and the corresponding sensor measurements, which may lead to false alarms if simple threshold-based monitoring is used. Fourth, the time evolution of faults can have a wide range of time scales, depending on the fault mechanism; switch faults will very quickly manifest, whereas degradation in a power source could take place over days or weeks. Our use of BNs and ACs, as discussed in this paper, is motivated by the need to construct EPS diagnostic models that capture both deterministic and uncertain behaviors when many modes are present.

III. BNs AND ACs

We now briefly present the underlying formalisms of our probabilistic model-based reasoning approach: BNs and ACs.

A. BNs

BNs represent multivariate probability distributions and are used for reasoning and learning under uncertainty [2]. Probability theory and graph theory form the basis of BNs: Roughly speaking, random variables are represented as nodes in a directed acyclic graph (DAG), whereas conditional dependencies are represented as graph edges. A BN, whose graph structure often reflects a domain's causal structure, is a compact representation of a joint probability table if its graph is relatively sparse. Both discrete and continuous random variables can be represented in BNs; our main emphasis in this paper is on BNs with discrete random variables. Each discrete random variable (or node) X has a finite number of states $\{x_1, \ldots, x_m\}$ and is parameterized by a conditional probability table (CPT).

Let $\boldsymbol{X}$ be the BN nodes, $\boldsymbol{E} \subset \boldsymbol{X}$ be the evidence nodes, and $\boldsymbol{e}$ be the evidence. Different probabilistic queries can now be formulated; they all assume that all nodes in $\boldsymbol{E}$ are clamped to values $\boldsymbol{e}$. Computation of most probable explanation (MPE) amounts to finding an MPE over the remaining nodes $\boldsymbol{R} = \boldsymbol{X} - \boldsymbol{E}$ or MPE$(\boldsymbol{e})$. Computation of marginals (or beliefs) amounts to inferring the posterior probabilities over one or more query nodes $\boldsymbol{Q} \subseteq \boldsymbol{R}$, specifically BEL$(Q, \boldsymbol{e})$, where $Q \in \boldsymbol{Q}$. Marginals are used to compute most likely values (MLVs) simply by picking, in BEL$(Q, \boldsymbol{e})$, a most likely state. Computation of the maximum *a posteriori* probability (MAP) generalizes MPE computation and finds a most probable instantiation over nodes $\boldsymbol{Q} \subseteq \boldsymbol{R}$, i.e., MAP$(\boldsymbol{Q}, \boldsymbol{e})$. MAP can be approximated by MPE and MLV, and we will denote this using $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{Q}, \boldsymbol{e})$ and $\mathrm{MAP}_{\mathrm{MLV}}(\boldsymbol{Q}, \boldsymbol{e})$, respectively. $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{Q}, \boldsymbol{e})$ is the result of disregarding the nodes in $\boldsymbol{R}$ not in $\boldsymbol{Q}$, and $\mathrm{MAP}_{\mathrm{MLV}}(\boldsymbol{Q}, \boldsymbol{e})$ is the result of aggregating MLV$(Q, \boldsymbol{e})$ of all $Q \in \boldsymbol{Q}$. These two approximations are of interest because of the greater computational complexity of MAP [13], compared with MPE and marginals [11], [12].

Different BN inference algorithms can be used to perform the preceding computations. We distinguish between exact and inexact algorithms and focus, in this paper, on exact algorithms, which include JTP [18]–[20], conditioning [21], [22], VE [23], [24], and ACE [5], [6]. In resource-bounded systems, including real-time avionics systems, there is a strong need to align the resource consumption of diagnostic computation with resource bounds [7], [8]. The compilation approach—including JTP and ACE—is attractive in resource-bounded systems. In this paper, we emphasize compilation into ACs, which we present next.

B. ACs

ACs, as discussed in [5] and [25], are used here to perform probabilistic inference. The compilation from BNs to ACs is based on the following connection between BNs and multilinear functions (MLFs). With each BN, we associate a corresponding MLF that computes the probability of evidence. For example, the BN $A \rightarrow C \leftarrow B$, where A and B are Boolean and C has three values, induces the following MLF:

$$\lambda_{a_1}\lambda_{b_1}\lambda_{c_1}\theta_{a_1}\theta_{b_1}\theta_{c_1|a_1,b_1} + \lambda_{a_1}\lambda_{b_1}\lambda_{c_2}\theta_{a_1}\theta_{b_1}\theta_{c_2|a_1,b_1} + \cdots$$
$$+\lambda_{a_2}\lambda_{b_2}\lambda_{c_2}\theta_{a_2}\theta_{b_2}\theta_{c_2|a_2,b_2} + \lambda_{a_2}\lambda_{b_2}\lambda_{c_3}\theta_{a_2}\theta_{b_2}\theta_{c_3|a_2,b_2}.$$

The terms in the MLF are in one-to-one correspondence with the rows of the network's joint distribution. Assume that all *indicator variables* λ_x have value 1 and all *parameter variables* $\theta_{x|\boldsymbol{u}}$ have value $\mathsf{Pr}(x|\boldsymbol{u})$. Each term will then be a product of probabilities, which evaluates to the probability of the corresponding row from the joint. The MLF will add all probabilities from the joint for a sum of 1.0. To compute the probability $\mathsf{Pr}(\boldsymbol{e})$ of evidence $\boldsymbol{e}$, we exclude certain terms from the sum. This is accomplished by carefully setting certain indicators to 0, instead of 1, according to the evidence.

Unfortunately, the network MLF has exponential size. However, if we can factor the MLF into something small enough to fit within memory, then we can compute $\mathrm{Pr}(\boldsymbol{e})$ in time that is linear in the size of the factorization. The factorization will take the form of an AC, which is a rooted DAG, where an internal node represents the sum or product of its children, and a leaf represents a constant or variable. In this context, those variables will be indicator and parameter variables. We refer to this process of producing an AC from a BN as *compiling* the network. While a BN is more compact than an AC, there are, in fact, several advantages associated with using an AC for probabilistic inference, as we will shortly discuss.

Once we have an AC for a network, we can compute $\mathrm{Pr}(\boldsymbol{e})$ for given evidence $\boldsymbol{e}$ by assigning appropriate values to leaves and then computing a value for each internal node in bottom–up fashion. The value for the root is then the answer to the query. We can also compute answers to many other queries (a posterior marginal for each network variable, a posterior marginal for each network family, etc.) by performing a second downward pass [5] analogous to the outward pass of the join tree algorithm. Hence, many queries can simultaneously be computed in time linear in the size of the AC. MPE$(\boldsymbol{e})$ may be computed in a similar manner by using maximization nodes, instead of addition nodes in the AC. Another main point is that the upward and downward passes may then be repeated for as many evidence sets as desired, without recompiling. Performing inference using an AC is therefore divided into two phases: 1) an offline phase, which compiles the network into an AC and is run once, and 2) an online phase, which answers many queries each time it is invoked and may be invoked multiple times.

We close this section by noting the close relationship between the JTP algorithm [18], [19] and ACs since the data structures involved in this algorithm embed an AC in a very precise sense [26]. Other compilation algorithms have been developed based on tabular elimination [25], weighted model counting [27], and ADD elimination [6]. These algorithms can have an exponential advantage over join tree by exploiting structure in the parameters of the BN [28].

TABLE III
SYNTAX OF OUR NOVEL SPECIFICATION LANGUAGE

```
<eps> ::= <component>+
<component> ::= (<source> | <basic> | <sensor> | <sink>) ";"
<source> ::= <name> ":" "source" ":" <p> ":"
<basic> ::= <name> ":" <btype> ":" <p> ":" <name>+
<sensor> ::= <name> ":" <stype> ":" <p> ":" <name>
<sink> ::= <name> ":" "sink" ":" <p> ":" <name>+
<btype> ::= "load" | "wire" | "inverter" | "breaker" | "relay"
<stype> ::= "sensorCurrent" | "sensorVoltage" | "sensorTouch"
```

IV. HIGH-LEVEL MODEL

Our approach to probabilistic model-based diagnosis involves four stages. In the first stage, we describe the EPS using a high-level modeling language. In the second stage, we apply a program to automatically convert the high-level specification into a BN. Putting the EPS model into the form of a BN allows us to leverage a large body of existing work on inference techniques. In the third stage, we compile the BN into an AC. This stage represents the application of a specific technique (ACs) for performing inference in BNs, which, in the resource-bounded real-time context, has significant advantages over other techniques [7], [8]. All stages up to this point have taken place offline, before the EPS is put into actual use. The fourth stage involves applying algorithms to the AC to perform inference online, when the EPS is in the field. By this time, as much computational effort as possible has been performed offline, leaving much less computation to be performed online. In this and the next four sections, we provide more detail on each stage, beginning in this section with the novel high-level specification language.

The syntax of our high-level specification language is given in Table III. A specification is a list of statements. (Here, $\langle$`name`$\rangle$ is an identifier, and $\langle$`p`$\rangle$ is a probability.) Each statement defines a component, which can either be a *source* (battery), a *basic* component, a *sensor*, or a *sink* (load). For brevity, we do not describe here some statements defining more complicated sensors. The general idea is that power flows from sources through basic components to sinks, monitored by sensors, and various failures can occur at each component. For each component, we define its name, its type (e.g., `source`, `load`, `breaker`, `relay`, `sensorCurrent`, `sensorVoltage`), the probability that the component will fail,[1] and a set of neighboring components. For a source, the set of neighbors is empty; for a basic component or a sink, we list all neighbors that lie between the component and a source of electricity; for a sensor, we list only the component to which the sensor is attached. These sets of neighbors serve to define the topology of an EPS.

Fig. 2 depicts a very simple example of an EPS, which is also described in Table IV by means of our specification language. The third line, e.g., defines a current sensor `curSens` with failure probability 0.0003 and attached to component `wire1` (which happens to be a wire defined in the second line). The fifth line defines a touch (or position) sensor `touchSens` with failure probability 0.0002 and attached to the relay `rly`.

[1] As described, all failures for a given component have equal probability, but the syntax can easily be extended to assign differing probabilities to different kinds of failures.

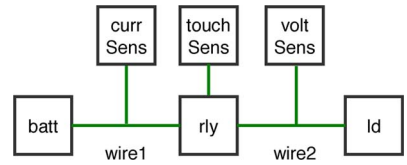


Fig. 2. Small EPS; it is described using our specification language in Table IV.

TABLE IV
SMALL EPS, SHOWN IN FIG. 2, DESCRIBED USING OUR SPECIFICATION LANGUAGE

batt :	source :	0.0001 ;	
wire1 :	wire :	0.0000 :	batt ;
curSens :	sensorCurrent :	0.0003 :	wire1 ;
rly :	relay :	0.0003 :	wire1 ;
touchSens :	sensorTouch :	0.0002 :	rly ;
wire2 :	wire :	0.0000 :	rly ;
voltSens :	sensorVoltage :	0.0002 :	wire2 ;
ld :	sink :	0.0001 :	wire2 ;

Using our specification language, the ADAPT EPS is described using statements for the following components: 3 sources (batteries), 20 sinks (loads), 16 wires (we only need to describe a wire if it has a sensor attached or if we want to model failures in wires), 2 inverters, 9 circuit breakers, 25 relays, 17 current and load sensors, 16 voltage sensors, 33 position (touch) sensors, and 6 more advanced sensors (these advanced sensors are not described here for the sake of brevity).

The main purpose of the high-level specification language is to make developing an EPS model easy and less error prone. One can specify a model by listing which components exist in the system and, for each, its type, failure probability, and neighbors. All of this information can often directly be obtained from schematics and hardware manuals. Consequently, the modeling task at the specification language level does not require guesswork or any knowledge of BNs or ACs.

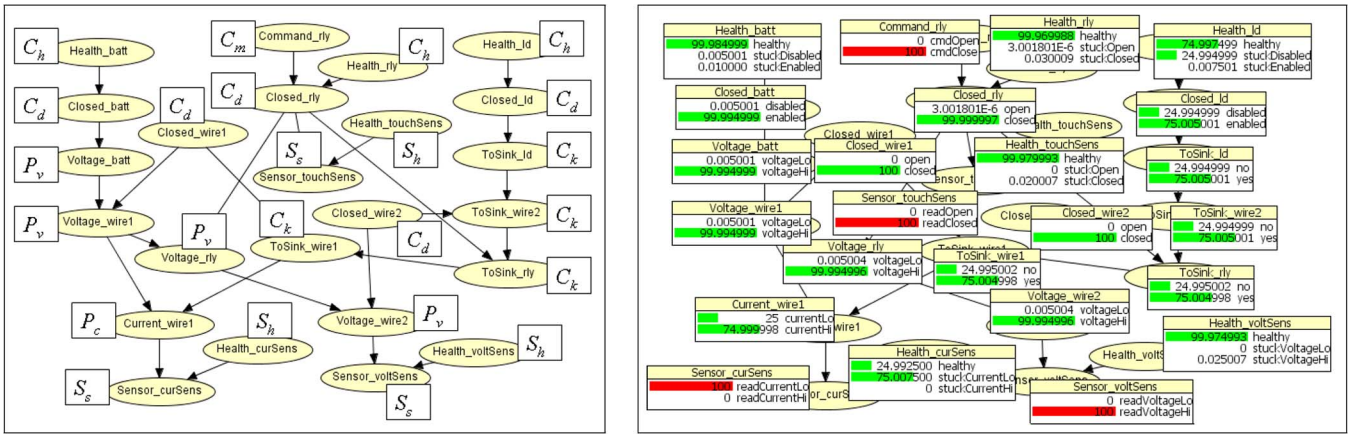
Components differ from each other in some ways that are not explicitly represented in the specification language, because the information can be inferred from the component's type. For example, some component types, such as a circuit breaker, accept a command to open or close, whereas other, such as a wire, do not. Similarly, different components may suffer different types of failures, as presented in Table I. For example, a wire can only fail in a stuck-open state, whereas a circuit breaker can be stuck open or stuck closed. This information is added during the BN autogeneration stage (see Section VI), reflecting our BN modeling approach, which is what we discuss next.

V. MODELING EPSs

A main contribution in this work is our systematic modeling of EPSs using BNs. BNs provide a probabilistic semantics for our high-level specification language, and in addition, they support efficient inference including compilation into ACs. We partition the set of BN nodes $\boldsymbol{X}$ into subsets $\boldsymbol{H}$, $\boldsymbol{E}$, $\boldsymbol{C}$, $\boldsymbol{P}$, and $\boldsymbol{R}$ as given here.

1) Health nodes ($\boldsymbol{H}$), where $\boldsymbol{H} = \boldsymbol{H}_C \cup \boldsymbol{H}_S$ and $\boldsymbol{H}_C \cap \boldsymbol{H}_S = \varnothing$. Here, $\boldsymbol{H}_C$ (component health nodes) represents the health of the EPS components, and $\boldsymbol{H}_S$ (sensor health nodes) represents the health of the EPS sensors.

TABLE V
(LEFT) BN AND (RIGHT) MPE COMPUTATION USING THE NETWORK FOR THE SMALL EPS SPECIFIED IN TABLE IV. IN THE BN, WE SHOW BOTH THE NODE NAMES ($Health_batt, Health_ld, \ldots$) AND THE NOTATION ($C_h, C_d, \ldots$) USED TO DESCRIBE THE AUTOGENERATION ALGORITHM



2) Evidence nodes ($\boldsymbol{E}$), where $\boldsymbol{E} = \boldsymbol{E}_C \cup \boldsymbol{E}_S$ and $\boldsymbol{E}_C \cap \boldsymbol{E}_S = \varnothing$. Here, $\boldsymbol{E}_C$ (command nodes) represents the commands to the EPS, whereas $\boldsymbol{E}_S$ (sensor reading nodes) represents sensor readings from the EPS.
3) Connection nodes ($\boldsymbol{C}$), where $\boldsymbol{C} = \boldsymbol{C}_R \cup \boldsymbol{C}_K$ and $\boldsymbol{C}_R \cap \boldsymbol{C}_K = \varnothing$. Here, $\boldsymbol{C}_R$ (source connection nodes) represents connection to a source (battery) in an EPS; $\boldsymbol{C}_K$ (sink connection nodes) represents connection to a sink (load) in an EPS.
4) Presence nodes ($\boldsymbol{P}$), where $\boldsymbol{P} = \boldsymbol{P}_C \cup \boldsymbol{P}_V$ and $\boldsymbol{P}_C \cap \boldsymbol{P}_V = \varnothing$. Here, $\boldsymbol{P}_V$ (voltage presence nodes) represents voltage, similar to water pressure, provided by a source (battery) in an EPS. $\boldsymbol{P}_C$ (current propagation or presence nodes) represents the flow, similar to water flow, of electrical current from a source (battery) to a sink (load) in an EPS. In our case, there is the presence of voltage iff there is a closed connection to one or more batteries; therefore, one may work with either $\boldsymbol{C}_R$ or $\boldsymbol{P}_V$.
5) Remaining EPS nodes ($\boldsymbol{R}$) are nodes that are not health, evidence, connection, or presence nodes. If $\boldsymbol{X}$ is the set of all BN nodes, then $\boldsymbol{R} = \boldsymbol{X} - \boldsymbol{H} - \boldsymbol{E} - \boldsymbol{C} - \boldsymbol{P}$.

The preceding node partitioning allows us to state different probabilistic queries of interest, discuss our EPS modeling approach using BNs (both the topology and the individual nodes associated with different EPS components), and clearly present the experimental protocol.

In Section III, we discussed, given query variables $\boldsymbol{Q} \subseteq \boldsymbol{X}$ and evidence $\boldsymbol{e}$, three probabilistic queries $\mathrm{MAP}(\boldsymbol{Q}, \boldsymbol{e})$, $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{Q}, \boldsymbol{e})$, and $\mathrm{MAP}_{\mathrm{MLV}}(\boldsymbol{Q}, \boldsymbol{e})$. By introducing the preceding partitioning, we can put $\boldsymbol{Q} = \boldsymbol{H}_C$, $\boldsymbol{Q} = \boldsymbol{H}_S$, or $\boldsymbol{Q} = \boldsymbol{H}$ and obtain a total of nine different diagnostics queries. As an example, $\boldsymbol{Q} = \boldsymbol{H}_S$ is of interest in sensor validation, where the main focus is on qualifying and disqualifying sensors [29], e.g., voltage sensors, current sensors, fuel sensors, or altitude sensors. In the rest of this paper, we emphasize $\boldsymbol{Q} = \boldsymbol{H}$ and particularly $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{H}, \boldsymbol{e})$.

A key contribution in this work is our modeling of EPSs using BNs. An EPS presents two different but closely related problems, i.e., a voltage differential problem and a current flow problem. For current to flow, there must be a voltage differential across the battery terminals. In addition, the EPS circuit needs to be closed, which typically happens when an EPS load is turned on and all other relays between the load and a battery are also closed. This bidirectional voltage–current propagation problem is different from and more complicated than the unidirectional flow problem posed by digital circuits implementing Boolean logic. Such digital electronic circuits have extensively been studied in the model-based diagnosis and BN literature [2].

Table V provides a simple example of our EPS modeling approach. This BN was autogenerated, as discussed in Section VI, from the specification in Table IV. Here, $\boldsymbol{H}_C = \{Health_batt, Health_ld\}$, and $\boldsymbol{H}_S = \{Health_curSens, Health_voltSens, Health_touchSens\}$; $\boldsymbol{E}_C = \{Command_relay\}$, and $\boldsymbol{E}_S = \{Sensor_curSens, Sensor_voltSens, Sensor_touchSens\}$. The topology of the ADAPT BN, which currently contains more than 500 nodes, is analogous to this BN's topology. A key point in this example is how the integration of voltage presence nodes ($\boldsymbol{P}_V = \{Voltage_batt, Voltage_wire1, Voltage_rly, Voltage_wire2\}$), sink connection nodes ($\boldsymbol{C}_K = \{ToSink_wire1, ToSink_rly, ToSink_wire2, ToSink_ld\}$), and the current flow node ($\boldsymbol{P}_C = \{Current_wire1\}$) helps solve the problems of voltage presence and current flow previously identified. Many nodes, including current flow nodes, can be pruned (and indeed have been here), because they are leaf nodes and not involved in sensors. Another key point is how sensors, e.g., the voltage sensor (nodes $Health_voltSens$ and $Sensor_voltSens$) and the current sensor (nodes $Health_curSens$ and $Sensor_curSens$), are integrated into the overall BN topology.

We now consider inference, as illustrated in Table V. Suppose that $\boldsymbol{e} = \{Command_rly = cmdClose, Sensor_curSens = readCurrentLo, Sensor_voltSens = readVoltageHi, Sensor_touchSens = readClosed\}$. This gives $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{H}, \boldsymbol{e}) = \{Health_batt = healthy,\ Health_ld = healthy, Health_curSens = stuckCurrentLo, Health_voltSens = healthy\}$. In words, if the command and sensor readings, except for $Sensor_curSens = readCurrentLo$, suggest that power is supplied to the load, then the MPE

diagnosis is that all components and sensors are healthy, except for the current sensor, where $Health_curSens = stuckCurrentLo$. It is reassuring that there is agreement between the MPE diagnosis and common sense in this case.

While our modeling approach can be used when manually constructing BNs for EPSs, it is even more powerful when automated, and we now turn to how we have formalized it in an autogeneration algorithm.

VI. Autogeneration of BN

In this section, we discuss how a BN is autogenerated from a high-level specification model. This is the second stage in our approach to probabilistic model-based diagnosis. The conversion runs in a loop, which processes one component from the specification in each iteration. Such a sequential order is guaranteed to exist under the assumption that the underlying EPS can be described using a DAG. There is a clear mapping from a high-level specification to a DAG: In each component statement (see Table III), the first ⟨name⟩ represents a node, and the ⟨name⟩+ part represents its parents (assuming this is not a source statement in the specification, which is trivial since it is a root node in the DAG). Under the assumption that there exists such a DAG, there exists a sequential high-level specification since it is well known from graph theory that any DAG can be topologically (or sequentially) sorted.

The autogeneration algorithm can now be summarized as follows: We iterate over the components in the specification and, for each, generate a set of BN nodes and a set of BN edges. Each time the algorithm creates a BN node for a component, it places the node into the appropriate set among $\boldsymbol{H}_C$, $\boldsymbol{H}_S$, $\boldsymbol{E}_C$, $\boldsymbol{E}_S$, $\boldsymbol{C}_R$, $\boldsymbol{C}_K$, $\boldsymbol{P}_C$, $\boldsymbol{P}_V$, and $\boldsymbol{R}$, as we illustrate here.

The processing of a sensor is somewhat different from the processing of other components, so we separately treat sensors, after first discussing other components. As an example, Fig. 3(a) depicts the part of a BN corresponding to a relay C. For the component C, the autogeneration algorithm generates six nodes in the BN.

1) A component health node $C_h \in \boldsymbol{H}_C$, with values $\{healthy, stuckOpen, stuckClosed\}$, indicates C's health state. C_h has a CPT set according to C's failure probability, as defined in the high-level specification.
2) A command node $C_m \in \boldsymbol{E}_C$, with values $\{cmdOpen, cmdClose\}$, indicates the command being sent to the relay. This value will always be known prior to inference since it is set according to the command being issued to the relay. Therefore, probabilities in this CPT are not important; C_m has a uniform CPT.
3) A remaining node $C_d \in \boldsymbol{R}$, with values $\{open, closed\}$, indicates whether C is currently closed. If $C_h = healthy$, then C_d indicates closed iff $C_m = cmdClose$. Otherwise, if C_h is $stuckOpen$ $(stuckClosed)$, then C_d indicates $open$ $(closed)$.
4) A source connection node $C_r \in \boldsymbol{C}_R$, with values $\{open, closed\}$, indicates whether there is a closed path from C to a battery (source). $C_r = closed$ iff $C_d = closed \wedge \vee_N[N_r = closed]$, where N iterates over all of C's upstream neighbors.[2]
5) A sink connection node $C_k \in \boldsymbol{C}_K$, with values $\{open, closed\}$, indicates whether there is a closed path from C to a load (sink). $C_k = closed$ iff $C_d = closed \wedge \vee_N[N_k = closed]$, where N iterates over all of C's downstream neighbors.
6) A current presence node $C_c \in \boldsymbol{P}_C$, with values $\{currentLo, currentHi\}$, indicates whether the current is flowing through C. $C_c = currentHi$ iff $C_r = closed$ and $C_k = closed$.

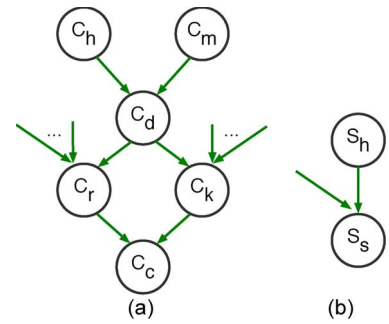


Fig. 3. Part of the BN corresponding to (a) a relay and (b) a current sensor.

For C_r and C_k, the disjunction is cascaded to prevent the CPT from becoming too large. This same template applies to all nonsensor components with a few minor modifications. For example, a source can set C_r to be equivalent to C_d; a sink can set C_k to be equivalent to C_d; a wire, which does not accept commands, will always set C_m to $cmdClosed$ (or omit C_m from the model); and different component types may have different types of failures.

Fig. 3(b) depicts the part of the BN corresponding to a current sensor S, which is attached to a node such as C_c of a component C. The autogeneration algorithm creates two nodes in the BN corresponding to S.

1) A sensor health node $S_h \in \boldsymbol{H}_S$, with values $\{healthy, stuckCurrentLo, stuckCurrentHi\}$, indicates S's health state. S_h has a CPT set according to S's failure probability, as defined in the high-level specification.
2) A sensor reading node $S_s \in \boldsymbol{E}_S$, with values $\{readCurrentLo, readCurrentHi\}$, indicates S's two-state discretized sensor reading. If $S_h = healthy$, then S_s indicates closed iff $C_c = currentHi$. Otherwise, if S_h is $stuckCurrentLo$ $(stuckCurrentHi)$, then S_s indicates $readCurrentLo$ $(readCurrentHi)$.

This same template applies to all sensor components (except some more complicated sensors, which are beyond the scope of this work) with a few minor modifications. For example, different sensors are attached to different nodes in C: current sensors are attached to C_c, voltage sensors are attached to C_r, and touch sensors are attached to C_d.

After the BN generation step previously discussed, there is a BN pruning step. Pruning takes place based on

[2] A neighbor of C is upstream of C if it is located between C and a source in the high-level specification. A neighbor of C is downstream of C if it is located between C and a sink in the high-level specification.

information about query nodes ($\boldsymbol{H}_C$ and $\boldsymbol{H}_S$), as well as about evidence nodes ($\boldsymbol{E}_C$ and $\boldsymbol{E}_S$). A common pruning technique involves removing leaf nodes that are not part of the evidence ($\boldsymbol{E}_C \cup \boldsymbol{E}_S$) or the query ($\boldsymbol{H}_C$, $\boldsymbol{H}_S$, or $\boldsymbol{H}_C \cup \boldsymbol{H}_S$) [30]. In Table V, some of the nodes have been pruned, compared to Fig. 3(a). Specifically, nodes corresponding to C_r, C_k, and C_c are pruned in Table V. In other words, for the relay shown in Table V, we have the following correspondence with the nonpruned nodes in Fig. 3(a): $C_h = Health_rly$, $C_m = Command_rly$, and $C_d = Closed_rly$. How can we determine to prune C_r, C_k, and C_c [referring to Fig. 3(a)] but not prune $S_s = Sensor_curSens$ and $S_h = Health_curSens$ (referring to Fig. 3(b) and Table V)? Here, $S_s = Sensor_curSens$ is a variable for which we assert evidence, whereas $S_h = Health_curSens$ is a query variable. Evidence and query variables are never pruned. We only prune nonevidence nonquery variables that are leaves or that become leaves as a result of other pruning. Consequently, all of C's nodes are pruned, except the following: $C_h = Health_rly$, which is a query variable; $C_m = Command_rly$, which is an evidence variable; and $C_d = Closed_rly$, which is neither, but cannot be pruned because it has a descendent that is an evidence variable, i.e., the touch sensor variable $Sensor_touchSens$.

A few assumptions have been made in our BN-generation approach. First, the approach assumes that a model can be expressed as a DAG since BNs are restricted to DAGs. Second, we do not model dynamic behavior (as induced, e.g., by capacitors or inductors) in the BN at this stage. Third, continuous sensor values are currently discretized into a small (ranging from two to four) number of states. However, the number of states could relatively easily be increased, or one could use soft evidence.

Our work is similar to existing work on constructing layered BNs with Noisy-MAX CPTs [31], [32]. For example, our component models are layered with a small number of layers, as illustrated in Fig. 3. The justification for our and similar research is the following. Even though existing WYSIWYG BN modeling tools, such as GENIE, HUGIN, and SAMIAM, are user friendly and intuitive to use, it still takes much effort and expertise to create BNs with hundreds or thousands of nodes. By using our approach, as discussed in Sections IV and V, the effort and level of expertise required to develop large-scale BNs are substantially reduced. While similar in spirit, there are also some differences between our and related research [31], [32]. For example, our autogenerated BNs do not have a fixed number of layers. Instead, the number of layers is determined by how component models are combined according to the structure of the system (see Table V).

VII. COMPILATION TO AC

We now very briefly summarize the compilation of BNs to ACs. Compilation is the third stage in our approach to probabilistic model-based diagnosis. Prior to compilation, we modify the BN's CPTs to store pointers to AC nodes rather than numbers. For example, if 0.1 is stored in a CPT slot, then this number would be replaced with a pointer to a single AC node (sink) labeled with 0.1. In addition, prior to compilation, for each BN variable, we add a new table over just that variable representing the values of that variable. For example, variable X with values 0 and 1 would generate a table over X, where the first slot contains a pointer to an AC node (sink) labeled with λ_0 and the second slot contains a pointer to an AC node (sink) labeled with λ_1.

After these two preprocessing steps, we run a slightly modified version of standard VE [24], [33]. The only difference occurs when the standard version wishes to add or multiply two numbers. In each of these situations, the standard algorithm will identify two slots A and B in tables, add (multiply) the two numbers residing there, and store the result back into some slot C of some table. When the modified algorithm looks into A and B, it finds pointers to AC nodes α and β rather than numbers. Instead of performing the arithmetic operation, the modified algorithm creates a new AC node γ labeled with "$+$" or "$*$," makes α and β children of γ, and stores a pointer to γ into C. Upon completion, standard VE yields a single table containing a single slot containing a number. The modified algorithm will be the same, except that, rather than a number, we will have a pointer to an AC node, which is the root of the compiled AC.

By exploiting local structure, this modified VE algorithm can yield an AC that is much smaller than exponential in treewidth. If one pays attention to how the CPTs of the BN representing EPSs are autogenerated, as described in Section VI, it is easy to see that many of these CPTs will be small and deterministic. AC compilation has been shown to perform well on many such BNs [6], [28], and the ADAPT BN is no exception.

VIII. EXPERIMENTAL RESULTS

We now discuss probabilistic inference experiments based on an ADAPT BN with 503 discrete nodes and 579 edges; related experiments can be found elsewhere [15], [16]. Probabilistic inference is the fourth and final stage in our probabilistic model-based diagnosis approach, and the only one that needs to be performed online. In the ADAPT BN, the number of states per node ranges from 2 to 4 with an average of 2.23 and a median of 2. Experimental data are divided into two sets: real-world data from ADAPT and synthetic data automatically generated from the ADAPT BN. For ACE, we used the ACE system to compile an ADAPT BN into an AC and to evaluate that AC (see http://reasoning.cs.ucla.edu/ace/). The timing measurements reported here were made on a personal computer with an Intel 4 1.83-GHz processor, 1-GB random access memory, and Windows XP.

A. Experiments using EPS Data

The purpose of the experiment with real-world data was to characterize the diagnostic quality of the ADAPT BN.

1) Design: For experimentation using real-world data, EPS scenarios were generated using the ADAPT EPS at NASA Ames (see https://dashlink.arc.nasa.gov/). These scenarios, which are summarized in Table VI, cover component failures, sensor failures, and both component and sensor failures. Each scenario contains one, two, or three faults. In order to stress-test our probabilistic reasoner, we did not restrict inserted faults to discrete faults only. We also inserted continuous faults, specifically faults of the form " stuck at x," "noise StdDev $= x$," or

TABLE VI
DIAGNOSTIC RESULTS FOR DIFFERENT FAULT SCENARIOS (WITH IDS 304, 305, . . .) FOR THE ELECTRICAL POWER SYSTEM TESTBED ADAPT

ID	Faults Inserted in ADAPT	Most Probable Diagnosis - Computed	Match
304	Relay EY260 failed open	*Health_relay_ey260_cl = stuckOpen*	Yes
305	Relay feedback sensor ESH175 failed open	*Health_relay_ey175_cl = stuckOpen*	Yes
306	Circuit breaker ISH262 tripped	*Health_breaker_ey262_op = stuckOpen*	Yes
308	Voltage sensor E261 failed low	*Health_e261 = stuckVoltageLo*	Yes
309	Battery BATT1 voltage low	*Health_battery1 = stuckDisabled*	Yes
310	Inverter INV1 failed off	*Health_inv1 = stuckOpen*	Yes
311	Light sensor LT500 failed low	*Health_lt500 = stuckLow*	Yes
441	Relay EY160 stuck open Big fan ST515 stuck at 0 RPM	*Health_relay_ey160_cl = stuckOpen*	Partly
442	Current sensor IT261 noise StdDev = 5 Relay feedback sensor ESH172 stuck at 0 Current sensor IT140 stuck at 100	*Health_it261 = stuckCurrentHi* *Health_esh172 = stuckOpen*	Partly
443	Current sensor IT281 drift slope = 2 Relay EY244 stuck closed Big fan ST516 stuck at -10 RPM	*Health_it281 = stuckCurrentHi* *Health_relay_ey244_cl = stuckClosed*	Partly
445	Voltage sensor E235 stuck at 0.3 Relay feedback sensor ESH344A stuck closed Inverter INV2 failed off	*Health_e235 = stuckVoltageLo* *Health_relay_ey344_cl = stuckClosed* *Health_inv2 = stuckOpen*	Partly
447	Voltage sensor E161 failed low Current sensor IT167 failed low	*Health_e161 = stuckVoltageLo* *Health_it167 = stuckCurrentLo*	Yes
449	Voltage sensor E140 failed low Voltage sensor E161 failed low	*Health_e140 = stuckVoltageLo* *Health_e161 = stuckVoltageLo*	Yes
450	Inverter INV1 failed off Big fan ST515 stuck at 600 RPM	*Health_inv1 = stuckOpen* *Health_fan1_speed_st515 = stuckMid*	Partly
451	Relay EY171 failed open Light sensor LT500 failed low	*Health_relay_ey171_cl = stuckOpen* *Health_lt500 = stuckLow*	Yes
452	Light bulb TE500 failed off Temperature sensor TE501 failed low	*Health_load170_bulb1 = stuckDisabled*	Partly

"drift slope $= x$," with $x \in \mathbb{R}$. Since our probabilistic models do not contain continuous random variables, experiments with continuous faults cannot be exactly diagnosed, but they are still of great interest.

In each scenario, ADAPT's initial state was given as follows: Circuit breakers were commanded closed; the corresponding command variables in $\boldsymbol{E}_C$ were clamped to $cmdClose$ in evidence $\boldsymbol{e}$. Relays were commanded open; the corresponding relay variables in $\boldsymbol{E}_C$ were clamped to $cmdOpen$ in $\boldsymbol{e}$. In this initial state, all health nodes $\boldsymbol{H}_E$ are deemed healthy when computing $\mathrm{MAP}(\boldsymbol{H}_E)$, $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{H}_E)$, or $\mathrm{MAP}_{\mathrm{MLV}}(\boldsymbol{H}_E)$. Continuous sensor readings were discretized before being used for clamping the corresponding discrete random variables $\boldsymbol{E}_C$ in our ADAPT model. To keep the experimental protocol consistent across scenarios, all inserted faults were persisted until the end of the experiments. Diagnostic queries $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{H}_E)$, for which results are presented in Table VI, were taken toward the end of each scenario.

2) Results: The results of the experiments with real-world data from ADAPT are summarized in Table VI. Each scenario is presented in one or more rows of the table, along with the faults inserted and the diagnostic results computed for queries $\mathrm{MAP}_{\mathrm{MPE}}(\boldsymbol{H}_E, \boldsymbol{e})$. Since $\boldsymbol{H}_E$ contains 128 variables, reflecting the health status of 128 EPS components and sensors, we only show the variables found to be nonhealthy in Table VI.

3) Discussion: We see in Table VI that the different diagnostic queries correctly diagnose a majority of these component and sensor failure scenarios. In fact, there is an exact match in 10 of the 16 scenarios. Even in cases where there is no exact agreement, the diagnosis is either partly matching or at least reasonable, as we will see in the following.

We now discuss in more detail experiments for which an exact match was not obtained. In Experiment 441 in Table VI, both EY160 and ST515 were failed. However, since EY160 is upstream of and controls the power to ST515, the nonperformance of ST515 is consistent with the single-fault diagnosis computed by ACE and in fact has a greater probability than the double faults actually inserted. In other words, the ST515 failure is masked by the EY160 failure.

Experiments 442 and 443 have continuous faults inserted that are currently beyond the scope of our discrete probabilistic model. In Experiment 442, there are no sensors on the affected load, making it difficult to detect whether 1) the relay has failed open, thus turning off the load, or 2) the relay feedback sensor has failed open. (Estimating how current varies for varying loads is beyond the scope of our discrete model.) So, if the relay failed open and turned off the attached load, there would be a drop in current being drawn from the battery, because there are fewer loads. However, we are not discretizing the current nodes in this way, sometimes making it difficult to distinguish between relay failure and relay position sensor failure.

In Experiment 443, two of the three faulty components were correctly isolated in spite of continuous faults being inserted. The one fault not caught was inserted in ST516, which is a fan on Load bank 2, which was not commanded on during this experiment. In other words, the fact that ST516 was neither on nor commanded to be on made the abnormally low sensor reading of -10 r/min harder to detect. Another issue was the discretization in the BN, where the faulty sensor reading of -10 is binned with the correct sensor reading of 0.

In Experiment 445, two of the three faulty components were correctly isolated, and the only difficulty was due to the continuous fault inserted.

In Experiment 450, two faults were inserted. When the inverter failed, all downstream power was disrupted. Thus, E165, E167, ST165, LT500, and IT167 go to low values. However, ST515, which should also have gone to a low value (because the fan no longer has power and therefore is not spinning), was stuck reading that its nominal value was around 600, which leads to the partly correct diagnosis $stuckMid$; the diagnosis $Health_inv1 = stuckOpen$ is correct.

In Experiment 452, the light sensor LT500 falls from ≈43 to ≈32 toward the end of the experiment. This lower value of ≈32 strongly suggests that only two bulbs are on or, in other words, that one bulb out of the three bulbs present had failed. Temperature sensor TE500 started falling, indicating that the bulb associated with that sensor was off. Then, TE501 went to 0, whereas the light sensor reading remained the same, indicating that sensor TE501 was likely also faulty. At the time of the diagnosis, we have the following evidence: TE500 reads *high* (however, its derivative is negative indicating that the bulb is off, but that is not in the discrete model); TE501 reads *low*; TE502 reads *high*; and LT500's sensor reading indicates that two bulbs are lit. Thus, based on the evidence provided to our model, it concludes a single fault of $stuckDisabled$ for Bulb 1. This diagnosis of $Health_load170_bulb1 = stuckDisabled$ is a direct result of the TE501 and LT500 readings. However, what was inserted was two faults, for Bulb 0 and TE501. This highlights two issues. First, the discretization does not perfectly capture the signature of a bulb being *off*. Here, the bulb is still warm from having previously been on, leading the TE500 value to be above the threshold defined for *on*. The second issue is that temporal aspects are not captured by taking one time slice near the end of the run; in this case, there are temporal clues that point toward the correct diagnosis.

We note that there are several different but related phenomena underlying the mismatches in Table VI. First, and reflecting the challenging nature of the fault scenarios that can be created using ADAPT, continuous faults (as inserted in experiments 441, 442, 443, 445, 448, and 450) are simply beyond the scope of our currently discrete probabilistic model. Second, there is no guarantee that the inserted faults are part of a unique MPE for the given evidence for $\boldsymbol{E}_E$. There may be multiple MPEs, or alternatively, m faults may have been inserted, but one or more explanations with $m-1$ faults or less turn out to be more probable. For example, three faults may have been inserted into ADAPT, but there is an explanation with two or one fault that has a higher or equal posterior probability. Experiment 441 is a good example of this effect. Third, there are faults that could have been detected had more fine-grained discretizations of random variables been used. Experiments 442 and 452 provide an example of this since the failure of the temperature sensor TE501 was quite dramatic and indicative of a sensor failure rather than only a failure in the light bulb TE500. A fourth phenomenon is that there might be too few, improperly placed, or inadequate sensors to distinguish between different faults. Many of the mismatches in these experiments could have been detected had more appropriate sensing been used; a detailed discussion of sensing issues is beyond the scope of this paper.

In summary, we have observed strong performance for our probabilistic model in these controlled experiments with ADAPT. We also note that a richer way of presenting diagnostic results would be helpful but nontrivial to provide. Specifically, it would be useful to have access to all nonzero explanations and their probabilities, not just the MPE but explanations with lower probabilities. These experimental results also motivate several other future research directions, as discussed in Section IX.

TABLE VII
RESULTS FOR DIFFERENT INFERENCE ALGORITHMS (VE, ACE, AND CTP) WHEN COMPUTING MPEs AND MARGINALS USING SYNTHETIC DATA GENERATED FROM THE ADAPT BN

Inference	MPE		Marginals	
Time (ms)	VE	ACE	JTP	ACE
Minimum	19.30	0.2235	9.792	0.5721
Maximum	40.21	2.5411	65.34	5.9228
Median	19.81	0.2260	10.52	0.6006
Mean	20.13	0.2625	11.01	0.7854
St. Dev.	1.554	0.2028	4.101	0.6970

B. Experiments Using Simulated Data

The goal of the experiment with synthetic data was to understand the performance of ACE versus alternative algorithms, particularly VE and JTP, in the ADAPT setting.

1) Design: In order to better understand the performance of ACE, we performed comparative experiments with VE and JTP. Simulated data were created by a program that 1) generated a set of failure scenarios according to the probabilities of the ADAPT BN's health nodes $\boldsymbol{H}_E$ and 2) generated evidence by doing stochastic simulation for each failure scenario. These evidence sets were then used as evidence in the three different inference systems, and inference was performed, as presented here.

2) Results: Results from the experiments are summarized in Table VII. Both MPEs and marginals were computed for 200 simulated evidence sets generated from the ADAPT BN.

3) Discussion: The main points, which are in line with previous results on a smaller version of the ADAPT BN [34], are as follows: On average, ACE is over 76 times faster than VE when computing MPEs (see Table VII). In addition, ACE can compute all marginals, supporting the probabilistic queries $\mathrm{BEL}(H, \boldsymbol{e})$ (where $H \in \boldsymbol{H}_E$) and $\mathrm{MAP}_{\mathrm{MLV}}\ (\boldsymbol{H}_E, \boldsymbol{e})$, using just slightly more than twice the time used for computing MPEs or $\mathrm{MAP}_{\mathrm{MPE}}\ (\boldsymbol{H}_E, \boldsymbol{e})$. In other words, ACE computes probabilities over 500 random variables more than 33 times faster than VE computes probabilities for a single random variable. The third inference system JTP can compute all marginals in a manner similar to ACE. This overcomes VE's limitation of computing probabilities for only one random variable at a time. Compared to ACE, however, JTP is over 14 times slower. Finally, the standard deviation is substantially smaller for ACE than for VE and JTP. The fast and predictable inference times of ACE are both very important factors for EPS health management in the real-time setting of aerospace.

Parametric structure can also be exploited by VE and JTP algorithms using more sophisticated representations of factors [35, Ch. 13]. However, the overhead associated with these techniques tends to outweigh the savings, unless the parametric structure is very excessive. The main benefit of using ACs is

that the overhead is pushed into the compilation phase and is factored out from the run-time process. This particular issue is discussed in some theoretical detail in [35], and there are also experimental results that illustrate this point.[3]

That said, it is clear that our approach produces, for ADAPT, a BN that all three systems (ACE, JTP, and VE) perform well on. This illustrates that the ADAPT BN was carefully constructed, using our novel modeling approach and autogeneration algorithm, in a manner that supports efficient inference using three quite different exact inference algorithms.

IX. Conclusion and Future Work

In this paper, we have discussed an EPS application of the probabilistic approach to model-based diagnosis. Specifically, we have discussed the use of BNs and ACs to perform diagnosis in EPSs in aircraft and spacecraft. We have emphasized two important issues that arise when developing diagnostic applications in this area, i.e., the challenges of modeling and real-time reasoning. The modeling challenge concerns how to model a real-world EPS by means of BNs. To address this challenge, we have developed a systematic way of representing EPSs as BNs, supported by an easy-to-use specification language and an autogeneration algorithm. The second challenge, that of real-time reasoning, is associated with the embedding of algorithms that solve computationally hard problems, including diagnostic reasoning, into hard real-time systems [7], [8]. To address this challenge, we compiled BNs into ACs.

While compilation of BNs to ACs is well established [6], [25]–[28], this paper further extends the reach of the technology by introducing a high-level EPS specification languages from which BNs are autogenerated, and showing that the combined approach gives strong experimental results on ADAPT, which is a real-world EPS.

Future directions of work include the following: First, improved modeling of and reasoning with continuous behavior, using soft evidence, highly discretized, and/or continuous random variables, along with representation using ACs for purposes of compilation, would be of great interest. A second area of interest is improved modeling of dynamic, transient, and cascading faults, along with their integration into the compilation approach. Third, it would be very useful to extend the high-level specification language and further investigate sensing issues, including the questions of optimal sensor placement, as well as the number and types of sensors needed to distinguish between different faults.

Acknowledgment

This material is based, in part, on work supported by NASA under Award NCC2-1426 and Award NNA07BB97C as well as NSF awards CCF0937044 and ECCS0931978. This work was performed while S. Uckun was at NASA. The authors would like to thank A. Patterson-Hine and D. Maclise (NASA ARC) for their central roles in the development of the ADAPT testbed, and D. Garcia and D. Nishikawa (NASA ARC) for generating the ADAPT data for many of our experiments.

[3](See http://www.ics.uci.edu/~csp/uai2006/tutorials#AdnanDarwiche).

References

[1] S. Poll, A. Patterson-Hine, J. Camisa, D. Garcia, D. Hall, C. Lee, O. J. Mengshoel, C. Neukom, D. Nishikawa, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos, "Advanced diagnostics and prognostics testbed," in *Proc. 18th Int. Workshop DX*, Nashville, TN, 2007, pp. 178–185.

[2] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[3] C.-F. Chien, S.-L. Chen, and Y.-S. Lin, "Using Bayesian network for fault location on distribution feeder," *IEEE Trans. Power Del.*, vol. 17, no. 3, pp. 785–793, Jul. 2002.

[4] Z. Yongli, H. Limin, and L. Jinling, "Bayesian network-based approach for power system fault diagnosis," *IEEE Trans. Power Del.*, vol. 21, no. 2, pp. 634–639, Apr. 2006.

[5] A. Darwiche, "A differential approach to inference in Bayesian networks," *J. ACM*, vol. 50, no. 3, pp. 280–305, 2003.

[6] M. Chavira and A. Darwiche, "Compiling Bayesian networks using variable elimination," in *Proc. 20th IJCAI*, Hyderabad, India, 2007, pp. 2443–2449.

[7] D. Musliner, J. Hendler, A. K. Agrawala, E. Durfee, J. K. Strosnider, and C. J. Paul, "The challenges of real-time AI," *Computer*, vol. 28, no. 1, pp. 58–66, Jan. 1995.

[8] O. J. Mengshoel, "Designing resource-bounded reasoners using Bayesian networks: System health monitoring and diagnosis," in *Proc. 18th Int. Workshop DX*, Nashville, TN, 2007, pp. 330–337.

[9] S. Singh, A. Kodali, K. Choi, K. R. Pattipati, S. M. Namburu, S. C. Sean, D. V. Prokhorov, and L. Qiao, "Dynamic multiple fault diagnosis: Mathematical formulations and solution techniques," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 1, pp. 160–176, Jan. 2009.

[10] S. Ruan, Y. Zhou, F. Yu, K. R. Pattipati, P. Willett, and A. Patterson-Hine, "Dynamic multiple-fault diagnosis with imperfect tests," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 6, pp. 1224–1236, Nov. 2009.

[11] F. G. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artif. Intell.*, vol. 42, no. 2/3, pp. 393–405, Mar. 1990.

[12] E. Shimony, "Finding MAPs for belief networks is NP-hard," *Artif. Intell.*, vol. 68, no. 2, pp. 399–410, Aug. 1994.

[13] J. D. Park and A. Darwiche, "Complexity results and approximation strategies for MAP explanations," *J. Artif. Intell. Res.*, vol. 21, no. 1, pp. 101–133, Jan. 2004.

[14] R. M. Button and A. Chicatelli, "Electrical power system health management," in *Proc. 1st Int. Forum Integr. Syst. Health Eng. Manage. Aerosp.*, Napa, CA, 2005.

[15] B. W. Ricks and O. J. Mengshoel, "The diagnostic challenge competition: Probabilistic techniques for fault diagnosis in electrical power systems," in *Proc. 20th Int. Workshop DX*, Stockholm, Sweden, 2009.

[16] O. J. Mengshoel, S. Poll, and T. Kurtoglu, "Developing large-scale Bayesian networks by composition: Fault diagnosis of electrical power systems in aircraft and spacecraft," in *Proc. IJCAI SAS—Reasoning and Integration Challenges*, 2009, pp. 59–66.

[17] D. R. Bromaghim, J. R. Leduc, R. M. Salasovich, G. G. Spanjers, J. M. Fife, M. J. Dulligan, J. H. Schilling, D. C. White, and L. K. Johnson, "Review of the electric propulsion space experiment (ESEX) program," *J. Propuls. Power*, vol. 18, no. 4, pp. 723–730, 2002.

[18] S. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *J. R. Stat. Soc. Ser. B*, vol. 50, no. 2, pp. 157–224, 1988.

[19] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen, "Bayesian updating in causal probabilistic networks by local computations," *SIAM J. Comput.*, vol. 4, pp. 269–282, 1990.

[20] P. P. Shenoy, "A valuation-based language for expert systems," *Int. J. Approx. Reason.*, vol. 5, no. 3, pp. 383–411, Sep. 1989.

[21] J. Pearl, "A constraint—Propagation approach to probabilistic reasoning," in *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer, Eds. Amsterdam, The Netherlands: Elsevier, 1986, pp. 357–369.

[22] A. Darwiche, "Recursive conditioning," *Artif. Intell.*, vol. 126, no. 1/2, pp. 5–41, Feb. 2001.

[23] Z. Li and B. D'Ambrosio, "Efficient inference in Bayes nets as a combinatorial optimization problem," *Int. J. Approx. Reason.*, vol. 11, no. 1, pp. 55–81, Jul. 1994.

[24] N. L. Zhang and D. Poole, "Exploiting causal independence in Bayesian network inference," *J. Artif. Intell. Res.*, vol. 5, no. 1, pp. 301–328, Aug. 1996.

[25] A. Darwiche, "A differential approach to inference in Bayesian networks," in *Proc. 16th Conf. UAI*, 2000, pp. 123–132.

[26] J. D. Park and A. Darwiche, "A differential semantics for jointree algorithms," *Artif. Intell.*, vol. 156, no. 2, pp. 197–216, Jul. 2004.

[27] A. Darwiche, "A logical approach to factoring belief networks," in *Proc. 8th Int. Conf. Principles Knowl. Representation Reason.*, 2002, pp. 409–420.
[28] M. Chavira and A. Darwiche, "Compiling Bayesian networks with local structure," in *Proc. 19th IJCAI*, 2005, pp. 1306–1312.
[29] O. J. Mengshoel, A. Darwiche, and S. Uckun, "Sensor validation using Bayesian networks," in *Proc. 9th iSAIRAS*, 2008.
[30] R. D. Shachter, "Evaluating influence diagrams," *Oper. Res.*, vol. 34, no. 6, pp. 871–882, Nov./Dec. 1986.
[31] P. Kraaijeveld and M. Druzdzel, "GeNIeRate: An interactive generator of diagnostic Bayesian network models," in *Proc. 16th Int. Workshop Principles Diagnosis*, 2005, pp. 175–180.
[32] K. Przytula, G. Isdale, and T.-S. Lu, "Collaborative development of large Bayesian networks," in *Proc. IEEE Autotestcon*, 2006, pp. 515–522.
[33] R. Dechter, "Bucket elimination: A unifying framework for reasoning," *Artif. Intell.*, vol. 113, no. 1/2, pp. 41–85, Sep. 1999.
[34] O. J. Mengshoel, A. Darwiche, K. Cascio, M. Chavira, S. Poll, and S. Uckun, "Diagnosing faults in electrical power systems of spacecraft and aircraft," in *Proc. 20th IAAI Conf.*, Chicago, IL, 2008, pp. 1699–1705.
[35] A. Darwiche, *Modeling and Reasoning With Bayesian Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

Ole J. Mengshoel (M'09) received the B.S. degree from the Norwegian Institute of Technology, Trondheim, Norway, in 1989, and the Ph.D. degree from the University of Illinois, Urbana, in 1999, both in computer science.

He is currently a Senior Systems Scientist with Carnegie Mellon University (CMU), Silicon Valley, CA, and affiliated with the Intelligent Systems Division, National Aeronautics and Space Administration (NASA) Ames Research Center, Moffett Field, CA. Prior to joining CMU, he was a Senior Scientist and Research Area Lead with USRA/RIACS and a Research Scientist with the Decision Sciences Group, Rockwell Scientific, and Knowledge-Based Systems, SINTEF, Norway. At NASA, he has a leadership role in the Diagnostics and Prognostics Group, Intelligent Systems Division, where his current research focuses on reasoning, machine learning, diagnosis, prognosis, and decision support under uncertainty—often using Bayesian networks—with aerospace applications of interest to NASA. He has published more than 35 papers and papers in journals, conference proceedings, and workshops. He is the holder of four U.S. patents.

Dr. Mengshoel is a member of the Association for the Advancement of Artificial Intelligence and the Association for Computer Machinery. He has served as a Reviewer and on program committees numerous times.

Mark Chavira received the B.S. degree from Loyola Marymount University, Los Angeles, in 1995, the M.S. degree in computer science from Stanford University, Stanford, CA, in 2001, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2007.

During his Ph.D. work, he specialized in the exact probabilistic inference subfield of artificial intelligence. Since 2007, he has been a Software Engineer with Google, Santa Monica, CA. At Google, he leads a team that applies probabilistic inference to create automated methods of understanding text. From 1995 to 2007, he was a Software Engineer with the Raytheon Systems Company, where he developed avionics operating system software and researched automatically routing unmanned aerial vehicles. His current research interests include logical and probabilistic inference.

Keith Cascio received the B.S. degree (*summa cum laude*) in computer science from Duke University, Durham, NC, in 2001.

Since 2002, he has been the Lead Developer of the Automated Reasoning Group, Department of Computer Science, University of California, Los Angeles, contributing to the SamIam system for modeling and reasoning with Bayesian networks.

Mr. Cascio is a member of Phi Beta Kappa.

Scott Poll received the B.S.E. degree in aerospace engineering from the University of Michigan, Ann Arbor, in 1994 and the M.S. degree in aeronautical engineering from the California Institute of Technology, Pasadena, in 1995.

He is currently a Research Engineer with the Intelligent Systems Division, National Aeronautics and Space Administration (NASA) Ames Research Center, Moffett Field, CA, where he is the Deputy Lead for the Diagnostics and Prognostics Group, Intelligent Systems Division. He is co-leading the evolution of a laboratory designed to enable the development, maturation, and benchmarking of diagnostic, prognostic, and decision technologies for system health management applications. He was previously the Associate Principal Investigator for Prognostics in the Integrated Vehicle Health Management Project in NASA's Aviation Safety Program.

Adnan Darwiche received the B.S. degree in civil engineering from Kuwait University, Kuwait City, Kuwait, in 1987, the M.S. and Ph.D. degrees in computer science from Stanford University, Palo Alto, CA, in 1989 and 1993, respectively.

He is currently a Professor and Chairman of the Department of Computer Science, University of California, Los Angeles (UCLA). Prior to joining UCLA in 1999, he was a Senior Scientist and Manager of the Department of Diagnostics and Modeling, Rockwell Science Center. He directs the automated reasoning group at UCLA, which is responsible for releasing some high-profile reasoning systems, including the Rsat satisfiability solver, the SamIam system for modeling and reasoning with Bayesian networks, and the c2d knowledge compiler. He is the Editor in Chief for the *Journal of Artificial Intelligence Research*. His research interests are symbolic and probabilistic reasoning and their applications to real-world problems.

Prof. Darwiche is a Fellow of the Association for the Advancement of Artificial Intelligence.

Serdar Uckun (S'89–M'95) was born in Izmir, Turkey. He received the M.D. degree in medicine from Ege University, Izmir, Turkey, the M.S. degree in biomedical engineering from Bogazici University, Istanbul, Turkey, the Ph.D. degree in biomedical engineering from Vanderbilt University, Nashville, TN, and the Postdoctoral Studies in computer science from Stanford University, Palo Alto, CA.

From 2004 to 2008, he served as Lead of the Discovery and Systems Health (DaSH) Technical Area, NASA Ames Research Center. Previously, he was the Director of the Research Institute for Advanced Computer Science (RIACS), from 2002 to 2004, the Director of Advanced Technology at Blue Pumpkin Software from 2000 to 2002, and the Assistant Director and Manager of the Intelligent Systems Department, Rockwell Science Center, Palo Alto Laboratory, from 1994 to 2000. He is currently the Manager of the Embedded Reasoning Area, Palo Alto Research Center, Palo Alto. He also serves as the President of the Prognostics and Health Management Society, which is a nonprofit educational and professional organization. He is an Associate Editor for the *International Journal on Prognostics and Health Management*. Previously, he has served as Associate Editor for the *Journal of Artificial Intelligence in Medicine* and on the Editorial Board of *Critical Reviews in Biomedical Engineering*. He is the holder of 12 U.S. patents on aerospace and enterprise systems technologies.

Dr. Uckun is a member of the Association for the Advancement of Artificial Intelligence, American Institute of Aeronautics and Astronautics, and Tau Beta Pi. He was the recipient of the NASA Exceptional Service Medal in 2006. Recently, he served as the General Chair of the 2008 International Conference on PHM.