Model-based Monitoring and Diagnosis of Systems with Software-Extended Behavior

Tsoline Mikaelian, Brian C. Williams, and Martin Sachenbacher
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
32 Vassar St., Room 32-275, Cambridge, MA, 02139
tsoline@mit.edu

Abstract

Model-based diagnosis has largely operated on hardware systems. However, in most complex systems today, hardware is augmented with software functions that influence the system's behavior. In this paper, hardware models are extended to include the behavior of associated embedded software, resulting in more comprehensive diagnoses. Prior work introduced probabilistic, hierarchical, constraint-based automata (PHCA) to allow the uniform and compact encoding of both hardware and software behavior. This paper focuses on PHCA-based monitoring and diagnosis to ensure the robustness of complex systems. We introduce a novel approach that frames diagnosis over a finite time horizon as a soft constraint optimization problem (COP), allowing us to leverage an extensive body of efficient solution methods for COPs. The solutions to the COP correspond to the most likely evolutions of the complex system. We demonstrate our approach on a vision-based rover navigation system, and models of the SPHERES and Earth Observing One spacecraft.

Introduction

Model-based diagnosis of devices has traditionally operated on hardware models (de Kleer & Williams 1987; Dressler & Struss 1996). For instance, given an observation sequence, the Livingstone (Williams & Nayak 1996) diagnostic engine estimates the state of hardware components based on hidden Markov models that describe each component's behavior in terms of nominal and faulty modes. At the other end of the spectrum, researchers have applied model-based diagnosis to software debugging (Mateis, Stumptner, & Wotawa 2000). This paper explores the middle ground between the two, in particular the monitoring and diagnosis of systems with combined hardware and software behavior.

Many complex systems today, such as spacecraft, robotic networks, automobiles and medical devices consist of hardware components whose functionality is extended or controlled by embedded software. Examples of devices with software-extended behavior include a communications module with an associated driver, and an inertial navigation unit with embedded software for trajectory determination. The embedded software in each of these systems interacts with the hardware components and influences their behavior. In order to correctly estimate the state of these devices, it is essential to consider their software-extended behavior.

As an example of a complex system, consider vision-based navigation for an autonomous rover exploring the surface of a planet. The camera in the navigation system is an instance of a device that has software-extended behavior: the image processing software embedded within the camera module augments the functionality of the camera by processing each image and determining whether it is corrupt. A sensor measuring the camera voltage may be used for estimating the physical state of the camera. A hardware model of the camera describes its physical behavior in terms of inputs, outputs and available sensor measurements. A diagnosis engine, such as Livingstone, that uses only hardware models will not be able to reason about a corrupt image. Consider, for instance, a scenario in which the camera sensor measures a zero voltage. Based solely on hardware models of the camera, the measurement sensor and the battery, the most likely diagnoses will include camera failure, low battery voltage and sensor failure. However, given a software-extended model of the camera that incorporates the behavior of the image processing software, the quality of the image may be used to correctly diagnose the navigation system. Given that the processed image is not corrupt, the most likely diagnosis, that the measurement sensor is broken, may be deduced. This scenario demonstrates that a diagnostic engine for systems with software-extended behavior must: 1) monitor the behavior of both the hardware and its embedded software, so that the software state can be used for diagnosing the hardware, and 2) reason about the system state, given delayed symptoms. An instance of a delayed symptom is the quality of the processed image.

In this paper we introduce a novel monitoring and diagnostic system that operates on software-extended behavior models, to meet requirements 1) and 2) listed above. In contrast to previous work on model-based software debugging, the purpose of this work is to leverage information within the embedded software to refine the diagnoses of physical systems. As such, we are not addressing the problem of diagnosing software bugs discovered at runtime, which can be handled by a separate exception handling mechanism. Capturing the behavior of software is more complex than that of hardware, due to the hierarchical structure of a program and the use of complex constructs. We address this complexity by using probabilistic, hierarchi-

cal, constraint-based automata (PHCA) (Williams, Chung, & Gupta 2001). Building upon this previous work, we introduce a PHCA-based monitoring and diagnostic engine that can handle delayed symptoms. While Livingstone-2 (L2) (Kurien & Nayak 2000) handles delayed symptoms for diagnosing hardware systems, our approach generalizes this capability to software-extended behavior, by posing the PHCA-based diagnosis problem as a receding finite horizon estimator. We frame diagnosis as a constraint optimization problem based on soft constraints that encode the structure and semantics of PHCA models. The soft constraint formulation provides convenient expressivity and enables the use of efficient optimal constraint solvers to enumerate the most likely diagnoses of the software-extended system.

Modeling Software-Extended Behavior

Consider the software-extended camera module for the vision-based navigation scenario introduced above. Assume that the failure probabilities for the battery, the camera and the sensor are 10%, 5% and 1% respectively. A typical behavioral model of the camera is shown on the left of Figure 1. The camera can be in one of 3 modes: on, off or broken. The hardware behavior in each of the modes is specified in terms of inputs to the camera, such as power, and the behavior of camera components, such as the shutter. The broken mode is unconstrained in order to accommodate novel types of failures. Mode transitions can occur probabilistically, or as a result of issued commands. The battery and the sensor can be modeled similarly. For the scenario introduced above, the most likely diagnoses of the module, generated based on the hardware models alone, are shown on the right of Figure 1. However, the unmodeled software components can offer important evidence that substantially alters the diagnosis. A sample specification of the behavior of the image processing software may take the following form:

> *If an image is taken by the camera, process it to determine whether it is corrupt. If the image is corrupt, discard it and reset the camera; retry until a non-corrupt image is obtained for navigation. Once a high quality image is stored, wait for new image requests from the navigation unit.*

Such a specification abstracts the behavior of the image processing software implemented in an embedded programming language, such as Esterel (Berry & Gonthier 1992) or RMPL (Williams, Chung, & Gupta 2001). For the above scenario, given that the image is not corrupt, the possibility that the camera is broken becomes very unlikely. This is illustrated in Figure 2. Unlike a hardware component that can typically be described by a single mode of behavior, monitoring software behavior necessitates tracking simultaneous hierarchical modes. Furthermore, a modeling framework for software-extended systems must support the specification of complex behavioral constructs. Probabilistic, hierarchical, constraint-based automata (PHCA) (Williams, Chung, & Gupta 2001) are compact encodings of hidden Markov models that capture both hardware and software behavior.

Definition 1 (PHCA)
A PHCA is a tuple $< \Sigma, P_\Theta, \Pi, O, C, P_T >$, where:

- Σ is a set of locations, partitioned into primitive locations Σ_p and composite locations Σ_c. Each composite location denotes a hierarchical, constraint automaton. A location may be marked or unmarked. A marked location represents an active execution branch.
- $P_\Theta(\Theta_i)$ denotes the probability that $\Theta_i \subseteq \Sigma$ is the set of start locations. Each composite location $l_i \subseteq \Sigma_c$ may have a set of start locations to be marked when l_i is marked.
- Π is a set of finite domain variables. $C[\Pi]$ is the set of all finite domain constraints over Π.
- $O \subseteq \Pi$ is the set of observable variables.
- $C : \Sigma \rightarrow C[\Pi]$ associates with each location $l_i \subseteq \Sigma$ a finite domain behavioral constraint $C(l_i)$.
- $P_T(l_i)$, for each $l_i \subseteq \Sigma_p$, is a probability distribution over a set of transition functions $T(l_i) : \Sigma_p^{(t)} \times C[\Pi]^{(t)} \rightarrow 2^{\Sigma^{(t+1)}}$. Each transition function maps a marked location into a set of locations to be marked at the next time step, provided that the transition's guard constraint is satisfied.

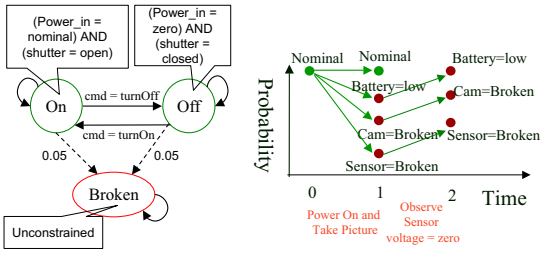


Figure 1: *left:* Behavior Model for the Camera Component. *right:* Most likely diagnoses based on hardware models.

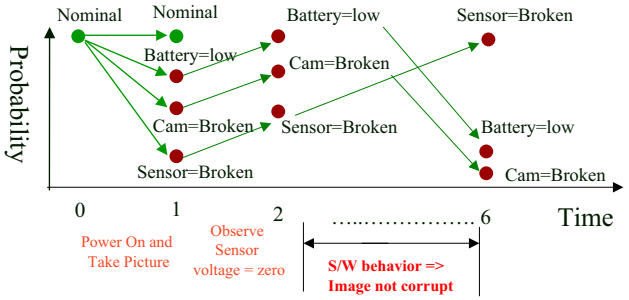


Figure 2: Most likely diagnoses based on software-extended behavior models. Nominal state = no failures.

Definition 2 (PHCA State)
The state of a PHCA at time t is a set of marked locations, called a marking $m^{(t)} \subset \Sigma$.

Figure 3 shows a PHCA model of the camera module. Circles represent primitive locations and boxes represent composite locations. The "On" composite location contains three subautomata that correspond to the primitive locations "Initializing", "Idle" and "Taking Picture". Each location of the PHCA may have behavioral constraints. In addition to the physical camera behavior, the model incorporates qualitative software behavior, such as processing the image. Furthermore, the possible camera configurations may be constrained by the embedded software. For example, if the image is determined to be corrupt, the software attempts to re-

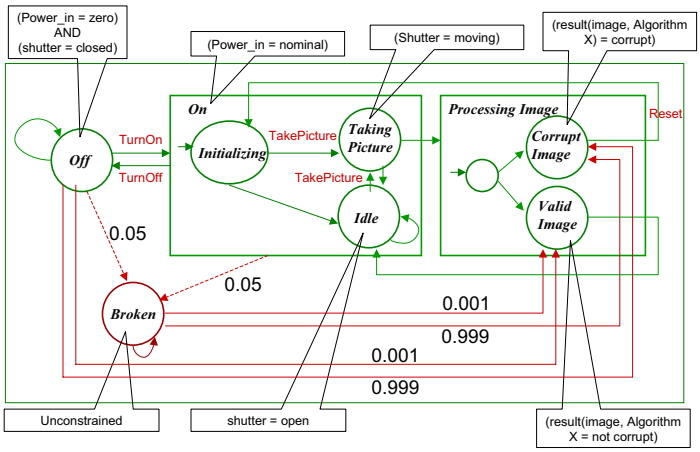


Figure 3: PHCA model for the camera/image processing module.

set the camera. This restricts the camera behavior to transition to the Initializing location.

To illustrate PHCA-based diagnosis, consider the state trajectories shown in Figure 2. At time step 2, as the sensor measurement indicates zero voltage, the most likely estimated trajectories end at 1) battery = low with 10% probability, 2) camera = broken with 5% probability and 3) sensor is broken with 1% probability. For the first trajectory, which indicates that the battery is low, the power to the camera is not nominal, hence the camera will stay in the "Off" location. For the second trajectory, the camera will be in the "Broken" location. For the third trajectory, which indicates that the sensor is broken, the power input to the camera will be unconstrained, and hence the PHCA state of the camera may include a marking of the "On" location. Although the evolutions of this third trajectory have an initially low probability of 1%, at time step 6 they become more likely than the others as the embedded software determines that the image is valid. The reason is because the second most likely trajectory at time 2 with camera = "Broken" location marked has a 0.001 probability of generating a valid image, thus making the probability of that trajectory 0.005% at time 6. This latter trajectory is less probable than those trajectories stemming from the sensor being broken with 1% probability. Similarly, the first trajectory with battery = low and camera = off becomes less likely at time step 6 as there is 0.001% probability of processing a valid image while the camera is "Off".

The following sections introduce a novel capability for PHCA-based trajectory tracking in the presence of delayed symptoms, addressing the requirements highlighted above.

Best-First Trajectory Enumeration for PHCA

PHCA-based monitoring and diagnosis are formulated as the task of enumerating and tracking the most likely trajectories of system state. Given a PHCA state distribution at time t and an assignment to observable and command variables in Π (see Definition 1) at times $t+1$ and t respectively, *Best-First Trajectory Enumeration* (BFTE) is the problem of estimating the most likely transitions to PHCA states at time $t+1$. Ideally, trajectory enumeration will maintain a complete probability distribution of all possible system trajectories. However, maintaining all possible state trajectories at each time step is intractable because of the exponential growth in state space. Thus at every time step, only a limited number of trajectories (K-Best) are maintained . A potential problem with this approach is that it may miss the best diagnosis if a trajectory through a pruned state that is initially very unlikely becomes very likely after additional evidence. For example, if we track only 3 trajectories in Figure 2, the initially unlikely state $(Sensor = Broken)$ at time 1 will be pruned, resulting in the best diagnosis to be unreachable when additional evidence is available at time 6.

Dealing with delayed symptoms is particularly important for diagnosing software-extended systems, due to typically delayed observations associated with complex processes. We generalize the L2 capability to PHCA-based diagnosis by performing BFTE within a receding N-Stage time horizon. We refer to this problem as *N-Stage Best-First Trajectory Enumeration* (N-BFTE) for PHCA. Although our approach still limits the number of tracked trajectories, it leverages the N-Stage history of observations and issued commands to reason about delayed symptoms.

Given a probability distribution $P(S^{(t)})$ of PHCA states S at time t, and assignments to observable and command variables in Π at times $(t \rightarrow t+N)$ and $(t \rightarrow t+N-1)$ respectively, the task of N-BFTE is to estimate the most probable trajectories, within the time horizon $(t \rightarrow t+N)$, that are consistent with the observations, commands and the PHCA model. We define the probability of a trajectory $\{S_i^{(t)}, S_{i+1}^{(t+1)}, S_{i+2}^{(t+2)}, .., S_{i+N}^{(t+N)}\}$ as:

$$P(S_i^{(t)}) \cdot \prod_{j=0..N-1} P_T(S_{i+j+1}^{(t+j+1)} | S_{i+j}^{(t+j)}, \Pi^{(t+j)}) \cdot \prod_{j=0..N} P(O^{(t+j)} | S_{i+j}^{(t+j)}) \quad (1)$$

where P_T is the transition probability from a current state to a target state. Since a PHCA state consists of multiple marked locations, each of which may transition to multiple target locations, T is a set of transitions taken from locations within the current state and leading to locations within the target state. Therefore, P_T is the product of the probabilities of all transitions $\tau \in T$. Within the consistency-based framework, the probability of observations $O^{(t+j)}$ for $j = 0..N$ in Equation 1 are given by:

$$P(O^{(t+j)} | S_{t+j}^{(t+j)}) = \begin{cases} 1 & if\ O^{(t+j)} \wedge S_{t+j}^{(t+j)}\ consistent \\ 0 & otherwise \end{cases}$$

Our approach to N-BFTE for PHCA models consists of two phases: an offline compilation phase and an online solution phase. In the offline phase, we frame the N-BFTE problem as a constraint optimization problem (COP) within the N-Stage horizon. In the online phase, the COP is solved to generate the K-Best PHCA trajectories in decreasing order of probability, as given by Equation 1. The following sections describe each phase in detail.

Offline Phase: Formulation of COP

We frame the N-BFTE problem for PHCA models as a soft constraint optimization problem (COP) (Schiex, Fargier, & Verfaillie 1995). The COP encodes the PHCA models as probabilistic constraints, such that the optimal solutions correspond to the most likely PHCA state trajectories. This formulation allows a separation between probability specification and variables to be solved for. Thus, we can associate probabilities with constraints that encode transitions, while solving for state variables.

Definition 3 (Constraint Optimization Problem) A *probabilistic constraint optimization problem* (COP) is a triple (X, D, F) where $X = \{X_1, ..., X_n\}$ is a set of variables with corresponding set of finite domains $D = \{D_1, ..., D_n\}$, and $F = \{F_1, ..., F_m\}$ is a set of functions $F_i : (S_i, R_i) \rightarrow [0, 1]$. Each constraint (S_i, R_i) consists of a scope $S_i = \{X_{i1}, ..., X_{ik}\} \subseteq X$, and a relation $R_i \subseteq D_{i1} \times ... \times D_{ik}$ defining all allowed assignments to variables in S_i. Each function F_i maps the tuples of (S_i, R_i) to a probability value in $[0, 1]$. Given variables of interest (solution variables) $Y \subseteq X$, a solution to the COP is an assignment to Y that is consistent with R, has a consistent extension to all variables X, and maximizes the global probability value in terms of the functions F_i.

We encode the structure and semantics of PHCA models as a probabilistic COP, consisting of:

- Set of variables $X_\Sigma^{(t)} \cup \Pi^{(t)} \cup X_{Exec}^{(t)}$ for $t = 0..N$, where $X_\Sigma^{(t)} = \{L_1^{(t)}, ..., L_n^{(t)}\}$ is a set of variables that correspond to PHCA locations $l_i \in \Sigma$, $\Pi^{(t)}$ is the set of PHCA variables at time t, and $X_{Exec}^{(t)} = \{E_1^{(t)}, ..., E_n^{(t)}\}$ is a set of auxiliary variables used to encode the execution semantics of the PHCA within the N-Stage time horizon.
- Set of finite, discrete-valued domains $D_{X_\Sigma} \cup D_\Pi \cup D_{X_{Exec}}$, where $D_{X_\Sigma} = \{Marked, Unmarked\}$ is the domain for each variable in X_Σ, D_Π is the set of domains for PHCA variables Π, and D_{Exec} is a set of domains for variables X_{Exec}. For brevity, we will abbreviate domain values throughout this paper as: CO=$Consistent$, IN=$Inconsistent$, MA=$Marked$, UM=$Unmarked$, DI=$Disabled$, EN=$Enabled$.
- Set of constraints R comprised of the behavioral constraints associated with PHCA locations, and an encoding of the PHCA execution semantics, presented below.
- Functions F that map tuples allowed by constraints R to probabilities. Disallowed tuples are assigned probability 0. For hard constraints, allowed tuples are assigned probability 1. For soft constraints, tuples are mapped to a range of probability values based on the PHCA model, as given by functions F presented below. These functions incorporate the probability distribution P_Θ of PHCA start states, and probabilities P_T associated with PHCA transitions.
- The optimal solutions to the COP are assignments to solution variables $X_\Sigma^{(t)}$ for $\{t..t + N\}$, representing the most probable PHCA state trajectories. The probability value of each trajectory is obtained by maximizing the product of the probabilities of the constraint tuples that are consistent with the observations and commands. This corresponds to maximizing the probability of the trajectory as given by Equation 1.

A key to framing PHCA-based N-BFTE as a COP is the formulation of the constraints R that capture the behavior and execution semantics of the PHCA. PHCA execution involves determining the consistency of behavioral constraints, identifying enabled transitions from a current PHCA state, and taking those transitions to determine the next state. Referring to the PHCA model in Figure 3, if we assume that at time t the PHCA state is $< On < Idle >>$ and that the transition guard constraint $(command = TakePicture)$ is satisfied, and at time t+1 the behavioral constraint $(shutter = moving)$ of the transition's target location is consistent, then the PHCA state at time t+1 will be $< On < TakingPicture >>$.

To encode the consistency of conditions, such as $(command = TakePicture)$, with observables and commands, a variable E_T is introduced with domain $\{CO, IN\}$ to denote whether the transition guard condition is consistent. Consistency is then formulated as a COP constraint that associates $E_T = CO$ with all possible assignments to the variable *command* that are consistent with the condition $(command = TakePicture)$. Consistency constraints are generated for all locations that have behavioral constraints and for all transitions that have guard constraints:

Behavioral Consistency:

$(\forall\ t \in \{0..N\},\ \forall\ L \in \Sigma : Behavior_L^{(t)}$=$CO \Leftrightarrow BehavioralConstraint(L)^{(t)})$.

Transition Guard Consistency:

$(\forall\ t \in \{0..N-1\},\ \forall\ \tau \in T : Guard_L^{(t)}$=$CO \Leftrightarrow GuardConstraint(\tau)^{(t)})$.

Some semantic rules apply to PHCA hierarchies. For example, when a composite location becomes marked, all of its start locations are attempted to be marked. We refer to this semantics as *Full Marking* of composite locations. The actual marking of locations depends on the consistency of each location's behavioral constraint with observations and commands. Since "Initializing" is a start location of the composite "On" location, a PHCA in state $< Off >$ may transition to state $< On < Initializing >>$. Furthermore, a composite location should be marked if any of its subautomata are marked, and unmarked if none of its subautomata are marked. We refer to this as *Hierarchical Composite Marking/Unmarking*. These rules are encoded by constraints.

The following four constraints apply only to the initial time $t = 0$, and will be referred to as Initial Constraints IC:

Initial Model Marking: enforces the initial marking of PHCA models: $(\forall C \in \{$*PHCA Models*$\} : C^{(0)}$=$MA)$.

Initial Unmarking: enforces the initial unmarking of all non-start locations:

$(\forall C \in \Sigma_C : (\forall L \in \{Subautomata(C)\text{-}\Theta(C)\} : L^{(0)}$=$UM))$.

Initial Full Marking: enables the initial full marking (marking of start locations) of composites:

$(\forall C \in \Sigma_C : [C^{(0)}{=}MA \Leftrightarrow (\forall L \in \Theta(C) : Start_L^{(0)}{=}EN)] \wedge [C^{(0)}{=}UM \Leftrightarrow (\forall L \in \Theta(C) : Start_L^{(0)}{=}DI)])$.

Initial Probabilistic Marking: encodes the initial probabilistic marking of start locations:

$(\forall C \in \Sigma_C : (\forall L \in \Theta(C) : [L^{(0)}{=}MA \Rightarrow (Start_L^{(0)}{=}EN \wedge Behavior_L^{(0)}{=}CO)]))$.

Each model M of the initial probabilistic marking constraint, with $Scope(M)=\{L^{(0)}, Start_L^{(0)}, Behavior_L^{(0)}\}$, is mapped to a probability value using the function:

$$F_0(M) = \begin{cases} Prob(L^{(0)} = MA) & if\ Condition1 \\ 1 - Prob(L^{(0)} = MA) & if\ Condition2 \\ 1.0 & otherwise \end{cases}$$

where $Condition1$ is:

$(L^{(0)}{=}MA) \wedge (Start_L^{(0)}{=}EN) \wedge (Behavior_L^{(0)}{=}CO)$,

and $Condition2$ is:

$(L^{(0)}{=}UM) \wedge (Start_L^{(0)}{=}EN) \wedge (Behavior_L^{(0)}{=}CO)$.

Hierarchical Composite Marking/Unmarking: encodes the hierarchical semantics that specifies how marking/unmarking of each composite location is affected by the state of its subautomata, as mentioned above. This constraint complements the full marking constraints that encode the semantics that specifies how marking of a composite location affects the marking of its subautomata. Hierarchical composite marking/unmarking is formulated as:

$(\forall t \in \{0..N\},\ \forall C \in \Sigma_C : [C^{(t)}{=}MA \Leftrightarrow (\exists L \in Subautomata(C) : L^{(t)}{=}MA)] \wedge [C^{(t)}{=}UM \Leftrightarrow (\forall L \in Subautomata(C) : L^{(t)}{=}UM)])$.

Probabilistic Transition Choice: encodes the probabilistic choice among transitions from each primitive location as an exclusive OR constraint, given by:

$(\forall t \in \{0..N-1\},\ \forall P \in \Sigma_P : (\exists \tau \in T \mid Source(\tau){=}P \Rightarrow [P^{(t)}{=}MA \Leftrightarrow (\exists T \in \{T|Source(T){=}P\} : T^{(t)}{=}EN \wedge (\forall T' \in (\{T|Source(T){=}P\}\text{-}\{\mathrm{T}\}) : T'^{(t)}{=}DI))] \bigwedge [P^{(t)}{=}UM \Leftrightarrow (\forall T \in \{T|Source(T){=}P\} : T^{(t)}{=}DI)]))$,

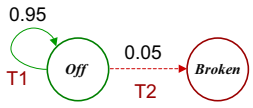
where each model M of this constraint, with $Scope(M) = \{P^{(t)}\} \cup \{T_i^{(t)} | Source(T_i) = P\}$, is mapped to a probability value using the function:

$$F_T(M) = \begin{cases} Prob(T_i) & if\ (\exists T_i^{(t)} : T_i^{(t)} = EN) \\ 1.0 & otherwise \end{cases}$$

The following example on the left of Figure 4 shows a probabilistic choice between two transitions for a section of the PHCA in Figure 3. In order to encode this, we first introduce a location variable $X_{Off}^{(t)}$, with domain $\{MA, UM\}$. Then auxiliary variables $E_{T1}^{(t)}$ and $E_{T2}^{(t)}$ with domain $\{EN, DI\}$ are introduced for transitions T1 and T2 respectively. The probabilistic transition choice constraint is instantiated at time t for the choice among T1 and T2:

$[X_{Off}^{(t)}{=}MA \Leftrightarrow (\exists T \in \{T1, T2\} : T^{(t)}{=}EN \wedge (\forall T' \in \{\{T1, T2\} - T\} : T'^{(t)}{=}DI))] \bigwedge [X_{Off}^{(t)}{=}UM \Leftrightarrow (\forall T \in \{T1, T2\} : T^{(t)}{=}DI)]$,

which is compiled into a set of allowed tuples M with associated probability values obtained using the function $F_T(M)$ above, as shown in Figure 4 (*right*).



$X_{Off}^{(t)}$	$E_{T1}^{(t)}$	$E_{T2}^{(t)}$	Prob.
Marked	Enabled	Disabled	0.95
Marked	Disabled	Enabled	0.05
Unmarked	Disabled	Disabled	1.0

Figure 4: *left:* PHCA with two probabilistic transitions. *right:* Probabilistic transition choice constraint.

Probabilistic choice is made among consistent transitions, as encoded by the following constraint.

Transition Consistency: encodes that marking of the transition's source location and the consistency of the transition's guard with issued commands are necessary conditions for a consistent transition:

$(\forall t \in \{0..N-1\},\ \forall \tau \in T : [\tau^{(t)}{=}EN \Rightarrow (Source(\tau)^{(t)}{=}MA \wedge Guard_\tau^{(t)}{=}CO)])$.

Target Identification: encodes whether there's any enabled transition(s) to each location:

$(\forall t \in \{0..N-1\}, \forall L \in \Sigma : [TransTo_L^{(t+1)}{=}EN \Leftrightarrow (\exists \tau \in \mathrm{T} \cap \{T|Target(T){=}L\} : \tau^{(t)}{=}EN)])$.

Target Full Marking: enables the full marking of composite targets:

$(\forall t \in \{1..N\}, \forall C \in \Sigma_C : [(\forall L \in \Theta(C) : Start_L^{(t)}{=}EN) \Leftrightarrow (TransTo_C^{(t)}{=}EN \vee Start_C^{(t)}{=}EN)] \bigwedge [(\forall L \in \Theta(C) : Start_L^{(t)}{=}DI) \Leftrightarrow (TransTo_C^{(t)}{=}DI \wedge Start_C^{(t)}{=}DI)])$.

The above constraints identify the enabled transitions and enable full marking of targets. The following two constraints encode taking enabled transitions.

Primitive Target Marking: formulates the marking of each primitive location, based on whether it is an identified target or an indirect target (i.e. start location of a composite), and whether its behavioral constraint is consistent:

$(\forall t \in \{1..N\}, \forall P \in \Sigma_P : [(P^{(t)}{=}MA) \Leftrightarrow (TranTo_P^{(t)}{=}EN \vee Start_P^{(t)}{=}EN) \wedge Behavior_P^{(t)}{=}CO] \bigwedge [(P^{(t)}{=}UM) \Leftrightarrow (TransTo_P^{(t)}{=}DI \wedge (Start_P^{(t)}{=}DI \vee Behavior_P^{(t)}{=}IN))])$.

Composite Target Marking: is a "weaker" version of the primitive target marking constraint, in order to allow consistency with the hierarchical composite marking/unmarking constraint.

$(\forall t \in \{1..\mathrm{N}\}, \forall C \in \Sigma_C : [C^{(t)}{=}MA \Rightarrow Behavior_C^{(t)}{=}CO] \bigwedge [C^{(t)}{=}MA \Leftarrow (TransTo_C^{(t)}{=}EN \vee Start_C^{(t)}{=}EN) \wedge Behavior_C^{(t)}{=}CO] \bigwedge [(C^{(t)}{=}UM) \Rightarrow (TransTo_C^{(t)}{=}DI \wedge (Start_C^{(t)}{=}DI \vee Behavior_C^{(t)}{=}IN))])$.

Online Phase: Best-First Trajectory Tracking

The N-BFTE problem, formulated in the offline phase as the COP presented above, is dynamically updated and solved in an online phase when new observations and commands are available. Assignments to observation and command variables are added to the COP as unary constraints with probability 1. We refer to these constraints within the N-Stage horizon as the history of observations H_O and the history of commands H_C. The N-BFTE solutions are the K most

N-BFTE (COP, $Traj_{(i=1..K)}^{(t\rightarrow t+N)}$, $H_O^{(t\rightarrow t+N)}$, $H_C^{(t\rightarrow t+N-1)}$) $\rightarrow Traj_{(i=1..K)}^{(t+1\rightarrow t+N+1)}$ ::

1. Update the COP (X, D, F):
   1.1 Shift the time horizon from $(t \rightarrow t+N)$ to $(t+1 \rightarrow t+N+1)$ by deleting observations $O^{(t)}$ and commands $C^{(t)}$ from constraints H_O and H_C.
   1.2 Add new observations $O^{(t+N+1)}$ and commands $C^{(t+N)}$ to the COP:
   $H_O^{(t+1\rightarrow t+N+1)}$=$O^{(t+N+1)} \cup H_O^{(t+1\rightarrow t+N)}$
   $H_C^{(t+1\rightarrow t+N)}$=$C^{(t+N)} \cup H_C^{(t+1\rightarrow t+N-1)}$
   1.3 Remove the initial constraints IC if $t = 0$, or the initial state constraint (see 1.4) if $t > 0$
   1.4 To track trajectories $Traj_{(i=1..K)}^{(t\rightarrow t+N)}$, constrain the initial states of $Traj_{(i=1..K)}^{(t+1\rightarrow t+N+1)}$ to be consistent with the states $S_{(i=1..K)}^{(t+1)} \in Traj_{(i=1..K)}^{(t\rightarrow t+N)}$, where each $S_{(i)}^{(t+1)}$ represents an assignment to solution variables $X_\Sigma^{(t+1)}$. Add to the COP the constraint:
   $(\forall\, L \in \Sigma : L^{(t+1)} \wedge (\bigvee_{i=1..K} S_i^{(t+1)}))$,
   with preference function F_I that maps each model M of this constraint to a probability value given by:
   $F_I(M) = P(S_i^{'(t)}) \cdot P_T(S_i^{(t+1)} | S_i^{'(t)}, \Pi^{(t)})$
   if $L^{(t+1)} \wedge S_i^{(t+1)}$ are consistent.
   The set of states $S_i^{'(t)}$) denote the beginning states of $Traj_{(i=1..K)}^{(t\rightarrow t+N)}$. The probabilities $P(S_i^{'(t)})$ are obtained using preference function F_0 if $t = 0$, or through previous iterations of N-BFTE otherwise.
2. Enumerate the K most probable trajectories by solving for sets of solution variables $\{X_\Sigma^{(t+1)}, ..., X_\Sigma^{(t+1+N)}\}$, using an optimal constraint solver.
3. Return $Traj_{(i=1..K)}^{(t+1\rightarrow t+N+1)}$, along with the probability associated with each trajectory, as given by Equation 1.

Figure 5: Online trajectory tracking pseudocode.

probable trajectories within the N-Stage horizon. The steps of the online process are given by the N-BFTE pseudocode in Figure 5.

As time progresses during the online solution phase, the N-stage horizon is shifted from $(t \rightarrow t+N)$ to $(t+1 \rightarrow t+N+1)$. In addition to incorporating new constraints for observations and commands, the COP over the new horizon is updated by constraining the start states at time $t+1$ to match the states of the trajectories within the previous horizon. Referring to Figure 2, if we consider a time horizon $(0 \rightarrow 6)$, even if we track $K = 3$ trajectories, the trajectory ending at state $(Sensor = Broken)$ at time 6 will have the highest probability based on the delayed observation. Consequently, the state $(Sensor = Broken)$ at time 2 will be maintained within the horizon $(0 \rightarrow 6)$. As the horizon is shifted to $(1 \rightarrow 7)$, the initial constraints IC for probabilistic marking of the start states are removed from the COP (part 1.3 of pseudocode). Instead, the trajectories $Traj_{(i=1..3)}^{(0\rightarrow 6)}$ are tracked by adding a constraint that limits the states at time 1 to correspond to those in $Traj_{(i=1..3)}^{(0\rightarrow 6)}$ (part 1.4 of pseudocode). The new constraint for initial states is a soft constraint that maps each initial state to its probability value. The probability of each initial state within the new horizon $(1 \rightarrow 7)$ is computed as the probability of the previous initial state multiplied by the transition probability to the new initial state, as given in the pseudocode.

Decreasing the number of trajectories K being tracked solves the delayed-symptom problem by maintaining a larger number of states at each time step. However, for a system with many combinations of similar failure states with high probability, the number of trajectories maintained will have to be very large in order to be able to account for a delayed symptom that supports an initially low probability state. For such systems, considering even a small number of previous time steps N gives enough flexibility to regenerate the correct diagnosis.

Implementation and Results

The PHCA model-based N-BFTE capability, described above, has been implemented in C++. Figure 6 shows both phases of the N-BFTE process. In the offline phase, the N-

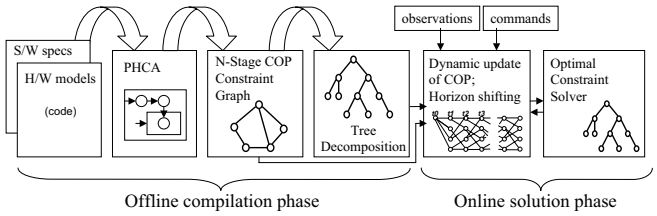


Figure 6: Process diagram for PHCA-based N-BFTE

Stage COP is generated automatically, given a PHCA model and parameter N. To enhance the efficiency of the solution phase, tree decomposition (Gottlob, Leone, & Scarcello 2000) is applied to decompose the COP into independent subproblems. This enables backtrack-free solution extraction during the online phase. In our implementation, the COP is decomposed using a tree decomposition package that implements bucket elimination (Kask, Dechter, & Larrosa 2003). The online monitoring and diagnosis process uses both the COP and its corresponding tree decomposition. The online phase consists of a loop that implements the N-BFTE pseudocode. At each iteration of the loop, the updated COP is solved using an implementation of the decomposition-based constraint optimization algorithm in (Sachenbacher & Williams 2004). For the camera model with $N = 6$, the COP has 436 variables and 441 constraints, and is solved online in $\sim$ 1.5 sec. We have further validated our system on models of two spacecraft:

1) SPHERES, a formation flying testbed for demonstrating novel metrology, control and autonomy technologies on the International Space Station (Nolet, Kong, & Miller 2004). We used two models of the SPHERES Global Metrology

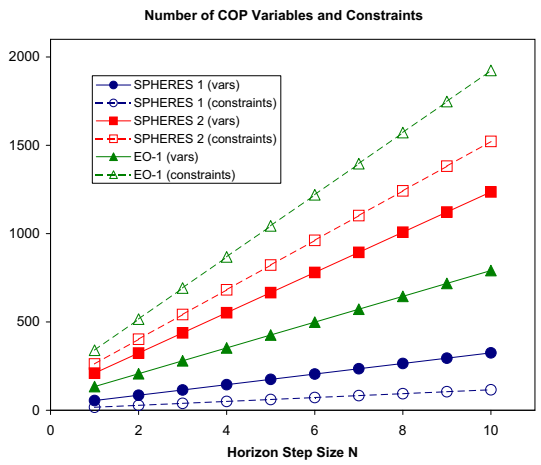


Figure 7: Size of the COP generated offline for the SPHERES and EO-1 models.

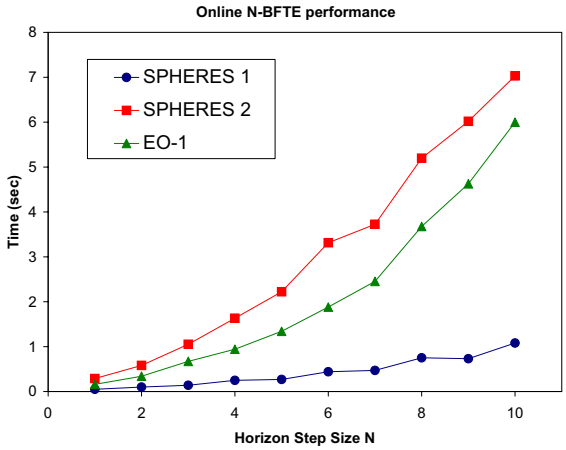


Figure 8: Online performance for the SPHERES and EO-1 models.

subsystem, which provides measurements to update estimates of position and attitude for each SPHERES satellite. The first model (SPHERES 1) consists of 5 components, and the second model (SPHERES 2) consists of 18 components.

2) Earth Observing One (EO-1), a New Millennium spacecraft which has validated a number of instrument and spacecraft technologies. Models of the EO-1 spacecraft were developed at NASA and used for validating L2 on EO-1 (Hayden, Sweet, & Christa 2004). The EO-1 model consists of 12 components, which are parts of the Advanced Land Imager, the Hyperion instrument and the Wideband Advanced Recorder Processor onboard the EO-1.

Both the SPHERES and EO-1 systems are modeled as probabilistic, concurrent constraint automata (PCCA) (Williams & Nayak 1996). A PCCA model is a special case of PHCA that does not support complex hierarchical constructs. Nevertheless, these models provided realistic scenarios for validating our approach. All experiments were run under Windows XP on a 1.6 GHz Pentium M processor. Figure 7 shows the size of the COP generated offline, for each of the models. Figure 8 shows the time required for solving the online N-BFTE problem. The results for the online phase were obtained for various nominal and failure scenarios as a function of the time horizon size N, for the same number of trajectories tracked (K=1 in Figure 8). For each of the tested scenarios, N-BFTE successfully generated the correct diagnosis based on delayed observations within the time horizon N. For instance, in the SPHERES scenarios most symptoms exhibited delays of 3 or 4 time steps.

Future work includes enhancing the efficiency of the online process by unifying decomposition-based and conflict-directed (Williams & Nayak 1996) techniques into the optimal constraint solver. Finally, investigating the optimal size of the diagnosis horizon and its relationship to the number of trajectories tracked will allow for automated parameter selection for the diagnostic system, based on properties of the modeled domain.

Acknowledgments

This research was sponsored in part by NASA CETDP under contract NNA04CK91A, and by the DARPA SRS program under contract FA8750-04-2-0243. The EO-1 models were provided by the NASA Ames Research Center.

References

Berry, G., and Gonthier, G. 1992. The *esterel* programming language: design, semantics and implementation. *Science of Computer Programming* 19(2):87–152.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130.

Dressler, O., and Struss, P. 1996. The consistency-based approach to automated diagnosis of devices. In *Proc. KR-96*, 267–311.

Gottlob, G.; Leone, N.; and Scarcello, F. 2000. A comparison of structural csp decomposition methods. *Artificial Intelligence* 124(2):243–282.

Hayden, S.; Sweet, A.; and Christa, S. 2004. Livingstone model-based diagnosis of earth observing one. In *Proc. AIAA Intelligent Systems*.

Kask, K.; Dechter, R.; and Larrosa, J. 2003. Unifying cluster-tree decompositions for automated reasoning. Technical report, U. of California at Irvine.

Kurien, J., and Nayak, P. 2000. Back to the future for consistency-based trajectory tracking. In *Proc. AAAI-00*.

Mateis, C.; Stumptner, M.; and Wotawa, F. 2000. Modeling java programs for diagnosis. In *Proc. ECAI-00*.

Nolet, S.; Kong, E.; and Miller, D. W. 2004. Autonomous docking algorithm development and experimentation using the spheres testbed. In *Proc. SPIE Defense and Security Symposium*.

Sachenbacher, M., and Williams, B. C. 2004. Diagnosis as semiring-based constraint optimization. In *Proc. ECAI-04*.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems:hard and easy problems. In *Proc. IJCAI-95*.

Williams, B. C., and Nayak, P. 1996. A model-based approach to reactive self-configuring systems. In *Proc. AAAI-96*, 971–978.

Williams, B. C.; Chung, S.; and Gupta, V. 2001. Mode estimation of model-based programs: monitoring systems with complex behavior. In *Proc. IJCAI-01*.