# Final Report

# Environmental Monitoring for Situation Assessment using Mobile & Fixed Sensors

**NASA-Ames Award No. NAG2-1337**

**Richard Fikes, Principal Investigator**

**Artificial Intelligence Laboratory**
**Computer Science Department**
**Stanford University**

**June 2004**

This project was co-led by Dr. Sheila McIlraith and Prof. Richard Fikes. Substantial research results and published papers describing those results were produced in multiple technology areas, including the following:

## Monitoring a Complex Physical System using a Hybrid Dynamic Bayes Net

The *Reverse Water Gas Shift system (RWGS)* is a complex physical system designed to produce oxygen from the carbon dioxide atmosphere on Mars. If sent to Mars, it would operate without human supervision, thus requiring a reliable automated system for monitoring and control. The RWGS presents many challenges typical of real-world systems, including: noisy and biased sensors, nonlinear behavior, effects that are manifested over different time granularities, and unobservability of many important quantities. In this portion of the project, we modeled the RWGS using a hybrid (discrete/continuous) *Dynamic Bayesian Network (DBN)*, where the state at each time slice contains 33 discrete and 184 continuous variables. We showed how the system state can be tracked using probabilistic inference over the model. We investigated how to deal with the various challenges presented by the RWGS, and produced a suite of techniques that are likely to be useful in a wide range of applications. In particular, we produced a general framework for dealing with nonlinear behavior using numerical integration techniques, extending the successful Unscented Filter. We also showed how to use a fixed-point computation to deal with effects that develop at different time scales, specifically rapid changes occurring during slowly changing processes. We tested our model using real data collected from the RWGS, demonstrating the feasibility of hybrid DBNs for monitoring complex real-world physical systems.

## A Formal Theory of Testing for Dynamical Systems

Just as actions can have indirect effects on the state of the world, so too can sensing actions have indirect effects on an agent's state of knowledge. In this portion of the project, we investigated "what sensing actions tell us", i.e., what an agent comes to know indirectly from the outcome of a sensing action, given knowledge of its actions and state constraints that hold in the world. To this end, we developed a formalization of the notion of testing within a dialect of the situation calculus that includes knowledge and sensing actions. Realizing this formalization required addressing the ramification problem for sensing actions. We formalized simple tests as sensing actions. Complex tests are expressed in the logic programming language Golog. We examined

what it means to perform a test, and how the outcome of a test affects an agent's state of knowledge. Finally, we developed automated reasoning techniques for test generation and complex-test verification, and precisely specified restrictions on when the techniques can be used. This work is relevant to a number of application domains including diagnostic problem solving, natural language understanding, plan recognition, and active vision.

### Diagnosing Hybrid Systems Using a Bayesian Model Selection Approach

In this portion of the project, we examined the problem of monitoring and diagnosing noisy complex dynamical systems that are modeled as hybrid systems; i.e., as systems having continuous behavior interleaved by discrete transitions. In particular, we examined continuous systems with embedded supervisory controllers that experience abrupt, partial, or full failure of component devices. We developed a mathematical formulation of the hybrid monitoring and diagnosis task as a Bayesian model tracking and selection problem, and developed a suitable tracking algorithm. The nonlinear dynamics of many hybrid systems present challenges to probabilistic tracking. Further, probabilistic tracking of a system for the purposes of diagnosis is problematic because the models of the system corresponding to failure modes are numerous and generally very unlikely. To focus tracking on these unlikely models and to reduce the number of potential models under consideration, we exploited logic-based techniques for qualitative model-based diagnosis to conjecture a limited initial set of consistent candidate models. We considered alternative tracking techniques that are relevant to different classes of hybrid systems, and focused specifically on a method for tracking multiple models of nonlinear behavior simultaneously using factored sampling and conditional density propagation. A motivating case study for this work was the problem of monitoring and diagnosing NASA's Sprint AERCam, a small spherical robotic camera unit with 12 thrusters that enable both linear and rotational motion.

## Publications

The publications listed below describe work that was performed solely or in part in this project. The first four of the listed papers are included in this final report.


U. Lerner, B. Moses, M. Scott, S. McIlraith, and D. Koller; "Monitoring a Complex Physical System using a Hybrid Dynamic Bayes Net"; Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2002); Edmonton, Canada; August, 2002.

S. McIlraith and R. Scherl; "What Sensing Tells Us: Towards a Formal Theory of Testing for Dynamical Systems"; Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'2000); pp. 483-490; August, 2000.

S. McIlraith; "Diagnosing Hybrid Systems: A Bayesian Model Selection Approach"; Proceedings of the Eleventh International Workshop on Principles of Diagnosis (DX'00); Morelia, Mexico; June 2000; pp. 140-146.

S. McIlraith, G. Biswas, D. Clancy, V. Gupta; "Hybrid Systems Diagnosis"; Proceedings of Hybrid Systems: Computation and Control; Lecture Notes in Computer Science; Springer-Verlag; pgs 282-295; 2000.

Todd Neller; "Simulation-Based Search for Hybrid System Control and Analysis"; Ph.D. Dissertation; Stanford University, Stanford, California, USA; June 2000.

# Monitoring a Complex Physical System using a Hybrid Dynamic Bayes Net

**Uri Lerner**
Computer Science Dept.
Stanford University
*uri@cs.stanford.edu*

**Brooks Moses**
Mechanical Engr. Dept.
Stanford University
*bmoses@stanford.edu*

**Maricia Scott**
Computer Science Dept.
Stanford University
*maricia@cs.stanford.edu*

**Sheila McIlraith**
Computer Science Dept.
Stanford University
*sam@ksl.stanford.edu*

**Daphne Koller**
Computer Science Dept.
Stanford University
*koller@cs.stanford.edu*

## Abstract

The *Reverse Water Gas Shift system (RWGS)* is a complex physical system designed to produce oxygen from the carbon dioxide atmosphere on Mars. If sent to Mars, it would operate without human supervision, thus requiring a reliable automated system for monitoring and control. The RWGS presents many challenges typical of real-world systems, including: noisy and biased sensors, nonlinear behavior, effects that are manifested over different time granularities, and unobservability of many important quantities. In this paper we model the RWGS using a hybrid (discrete/continuous) *Dynamic Bayesian Network (DBN)*, where the state at each time slice contains 33 discrete and 184 continuous variables. We show how the system state can be tracked using probabilistic inference over the model. We discuss how to deal with the various challenges presented by the RWGS, providing a suite of techniques that are likely to be useful in a wide range of applications. In particular, we describe a general framework for dealing with nonlinear behavior using numerical integration techniques, extending the successful Unscented Filter. We also show how to use a fixed-point computation to deal with effects that develop at different time scales, specifically rapid changes occurring during slowly changing processes. We test our model using real data collected from the RWGS, demonstrating the feasibility of hybrid DBNs for monitoring complex real-world physical systems.

## 1 Introduction

The *Reverse Water Gas Shift System (RWGS)* shown in Fig. 1 is a complex physical system designed and constructed at NASA's Kennedy Space Center to produce oxygen from carbon dioxide. NASA foresees a number of possible uses for the RWGS, including producing oxygen from the atmosphere on Mars and converting carbon dioxide to oxygen within closed human living quarters.

In a manned Mars mission, the RWGS would operate for 500 or more days without human intervention [Larson and Goodrich, 2000]. This level of autonomy requires the development of robust and adaptive software for fault diagnosis and control. In this paper, we focus on two key subtasks — *monitoring* and *prediction*. Monitoring, or tracking the current state of the system, is a crucial component

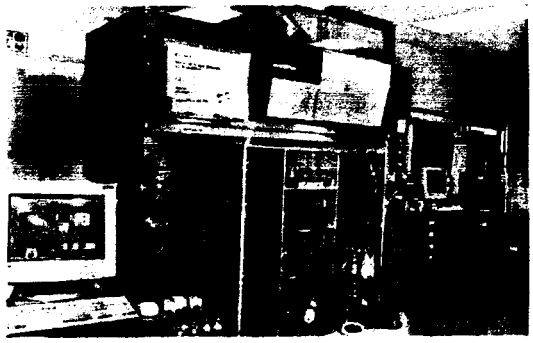


Figure 1: The Prototype RWGS System

of the control system. Prediction of the system's expected behavior is a basic tool in fault diagnosis — discrepancies between the predicted and the actual behavior of the system may indicate the presence of faults.

The RWGS presents a number of significant modeling and algorithmic challenges. From a modeling perspective, the system is very complex, and contains many subtle phenomena that are difficult to model accurately. Various phenomena in the system manifest themselves over dramatically different time scales, ranging from pressure waves that propagate on a time scale of milliseconds to slow changes such as gas composition that take hours to evolve. From a tracking perspective, the system dynamics are complex and highly nonlinear. Furthermore, the sensors give only a limited view of the system state. Some key quantities of the system are not measured, and the available sensors are noisy and biased, with both the noise level and the bias varying with the system state.

In this paper we model the RWGS using a hybrid (discrete/continuous) *Dynamic Bayesian Network (DBN)*, and show how the system state can be tracked using probabilistic inference over the model. We focus on the continuous part of the model, assuming all the discrete variables are known. We discuss how to deal with the various challenges presented by the RWGS, both in terms of modeling and in terms of inference. We provide a suite of techniques that are likely to be useful in a wide range of applications, in-

cluding the case where the discrete variables are not observed.

Perhaps the most interesting modeling problem presented by the RWGS is the issue of different time granularities. A naive solution is to discretize time at the finest granularity. Unfortunately, this approach is generally infeasible both because of the computational burden and because the number of observations is effectively reduced to one for every few thousand time steps, leading to serious inaccuracies. Instead, we take the approach of modeling the system at the time granularity of the observations. We show how to deal with the almost instantaneous changes relative to our time discretization by modeling a part of our system as a set of fixed-point equations.

For the inference task, we provide some new insights into the problem of tracking nonlinear systems. This task is commonly performed using the *Extended Kalman Filter (EKF)* [Bar-Shalom *et al.*, 2001] or the simpler and more accurate *Unscented Filter (UF)* [Julier and Uhlmann, 1997]. We view the problem as a numerical integration problem and demonstrate that the UF is an instance of a numerical integration technique. More importantly, our approach naturally leads to important generalizations of the UF: We show how to take advantage of the structure of the DBN and present a spectrum of filters, trading off accuracy with computational effort.

We tested our model using real data collected from the RWGS prototype system. Our results demonstrate the potential of using hybrid DBNs as a monitoring tool for complex real-world physical systems.

## 2 Preliminaries

In this paper, we characterize physical systems as discrete-time stochastic processes. System behavior is described in terms of a system state which evolves stochastically at discrete time steps $t = 0, 1, 2, \ldots$ We assume that the system is *Markovian* and *stationary*, i.e., the state of the system at time $t+1$ only depends on its state at time $t$, and the probabilistic dependencies are the same for all $t$.

The system state is modeled by a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. We partition the state variables $\mathcal{X}$ into a set of evidence (observed) variables, $\boldsymbol{E}$, and a set of hidden (unobserved) variables, $\boldsymbol{H}$. Physical systems commonly comprise both continuous quantities (e.g., flows, pressures, gas compositions) and discrete quantities (e.g., valve open/closed, compressor on/off). Consequently, we model such systems as *hybrid systems*, with $\mathcal{X}$ comprising both discrete and continuous variables.

We model the process dynamics of our system using a *Dynamic Bayesian Network (DBN)* [Dean and Kanazawa, 1989]. A DBN is represented as a Bayes Net fragment called a 2TBN, which defines the *transition model* $P(\boldsymbol{X}' \mid \boldsymbol{X})$ where $\boldsymbol{X}' = \{X_1, \ldots, X_m\}$ denotes the variables at time $t+1$ and $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ denotes some subset of the variables at time $t$ which are *persistent*, in that their values directly influence the next state. More formally, a DBN is a directed acyclic graph, whose nodes are random variables in two consecutive time slices, $\boldsymbol{X}$ and $\boldsymbol{X}'$. The edges in the graph denote direct probabilistic influence between the parents and their child. For every variable $X'$ at time $t+1$ we denote its parents as $\text{Par}(X') \subset \boldsymbol{X} \cup \boldsymbol{X}'$. Each $X'$ is also annotated with a *Conditional Probability Distribution (CPD)*, that defines the local probability model $P(X' \mid \text{Par}(X'))$. In our hybrid model, discrete nodes do not have continuous nodes as parents.

The *tracking* problem in DBNs is to find the *belief state* distribution $\text{Bel}(\boldsymbol{X}^t) \stackrel{def}{=} P(\boldsymbol{X}^t \mid e^1, \ldots, e^t)$, where $\boldsymbol{X}^t$ typically consists of the persistent variables $\boldsymbol{X}$ at time $t$, and $e^1, \ldots, e^t$ are the evidence variables from time 1 to time $t$. The belief state summarizes our beliefs about the state of the system at time $t$, given the observations from time 1 to time $t$. As such, it makes current and future predictions independent of past data. The tracking algorithm is an iterative process that propagates the belief state. We start with the belief state at time $t$, $\text{Bel}(\boldsymbol{X}^t)$ and perform three steps. We first compute $P(\boldsymbol{X}^t, \boldsymbol{X}^{t+1} \mid e^1, \ldots, e^t)$ as the product $\text{Bel}(\boldsymbol{X}^t)P(\boldsymbol{X}^{t+1} \mid \boldsymbol{X}^t)$. Next we marginalize out $\boldsymbol{X}^t$ resulting in a distribution over $\boldsymbol{X}^{t+1}$. Finally, we condition on $e^{t+1}$, and the result is the belief state at $t+1$, $\text{Bel}(\boldsymbol{X}^{t+1})$.

*Linear models* are an important class of DBNs. In a linear model, all the variables in $\boldsymbol{X}$ are continuous and all the dependencies are linear with some added Gaussian noise. More precisely, if a node $X$ has parents $Y_1, \ldots, Y_k$ then $P(X \mid Y_1, \ldots, Y_k) = \sum_{i=1}^{k} w_i Y_i + V$, where the $w_i$'s are constants and $V$ has a normal distribution $\mathcal{N}(\mu, \sigma^2)$. In a dynamic linear model, tracking can be done using a *Kalman filter* [Kalman, 1960], where the belief state is represented parametrically as a multivariate Gaussian in terms of the mean vector and the covariance matrix. Kalman filters therefore allow a compact belief state representation, which can be propagated in polynomial time and space.

When the dependencies in the model are nonlinear, the resulting distributions are generally non-Gaussian and cannot be represented in closed form. Consequently, the belief state is generally approximated as a multivariate Gaussian that preserves the first two moments of the true distribution. The traditional method for doing this approximation is using an *Extended Kalman Filter (EKF)* [Bar-Shalom *et al.*, 2001]. Assume that $\boldsymbol{X}' = f(\boldsymbol{X})$, where $f$ is some nonlinear function and $\boldsymbol{X} \sim \mathcal{N}(\mu, \Sigma)$. Note that we can always assume that $f$ is deterministic: If the dependency between $\boldsymbol{X}$ and $\boldsymbol{X}'$ is stochastic we can treat the stochasticity as extra random variables that $f$ takes as arguments. The EKF finds a linear approximation to $f$ around the mean of $\boldsymbol{X}$, i.e., we approximate $f$ using the first-order Taylor series expansion around $\mu$. The result is the linear function $f(\boldsymbol{X}) \approx f(\mu) + \nabla f|_{\mu}(\boldsymbol{X} - \mu)$, where $\nabla f|_{\mu}$ is the gradient of $f$ evaluated at $\mu$.[1]

[1] A second-order EKF approximation exists, but its increased complexity tends to limit its use.

The EKF has two serious disadvantages. The first is its inaccuracy — the EKF is accurate only if the second and higher-order terms in the Taylor series expansion are negligible. In many practical situations, this is not the case and using the EKF leads to a poor approximation. The second disadvantage is the need to compute the gradient. Some nonlinear functions may not be differentiable (e.g., the max function), preventing the use of an EKF. Even if the function is differentiable, computing the derivatives may be hard if the function is represented as a black box rather than in some analytical form.

The *Unscented Filter (UF)* [Julier and Uhlmann, 1997] provides an alternative approach to tracking nonlinear behavior. As with the EKF, the UF assumes that $X' = f(X)$ and $X \sim \mathcal{N}(\mu, \Sigma)$. The UF works by *deterministically* choosing $2d+1$ points $x_0, \ldots, x_{2d}$, where $x_0 = \mu$ and the other points are symmetric around $\mu$ (the actual points depend on $\Sigma$). Associated with each point is a weight $w_i$. The UF computes $x_i' = f(x_i)$ for $i = 0, 1, \ldots, 2d$, resulting in $2d+1$ points in $\mathbb{R}^m$, from which it estimates the first two moments of $X'$ as a weighted average of the $x_i'$'s. In particular, the mean $\mathcal{E}[X']$ is approximated as $\sum_{i=0}^{2d} w_i x_i'$.

The UF has several significant advantages over the EKF. First, it is easier to implement and use than the EKF — no derivatives need be computed, and the function $f$ is simply applied to $2d+1$ points. Second, despite its simplicity, the UF is more accurate than the EKF: The UF is a third-order approximation, i.e., inaccuracies are induced only by terms of degree four or more in the Taylor series expansion. Finally, instead of just ignoring the higher-order terms, the UF can account for some of their effects, by tuning a parameter used in the point selection. As shown in [Julier and Uhlmann, 1997], the UF can be extremely accurate, even in cases where the EKF leads to a poor approximation.

## 3 The RWGS System

The purpose of the RWGS is to decompose carbon dioxide ($CO_2$) (abundant on Mars) into oxygen ($O_2$) and carbon monoxide (CO). The system, shown in Fig. 2(a) [Goodrich, 2002], comprises two loops: a gas loop that converts $CO_2$ and hydrogen ($H_2$) into $H_2O$ and CO, and a water loop that electrolyzes the $H_2O$ to produce $O_2$ and $H_2$. Under normal operation, $CO_2$ at line (1) is combined with $H_2$ returned from the electrolyzer via line (12), and a mixture of $CO_2$, $H_2$, and CO from the reactor recycle line (11). This mixture enters a catalyzed reactor (3) heated to 400°C. Approximately 10% of the $CO_2$ and $H_2$ react to form CO and $H_2O$:

$$CO_2 + H_2 \rightleftharpoons CO + H_2O$$

The $H_2O$ is condensed at (4) and is stored in a tank (5). The remaining gas mixture passes through a separation membrane (9), which sends a fraction of the CO to the vent (13) while directing the remaining mixture into the recycle line (11). A compressor (10) is used to maintain the necessary pressure differential across the membrane. In the water loop, the $H_2O$ in tank (5) has some $CO_2$ dissolved in it, which would be detrimental to the electrolyzation process. To remedy this, the $H_2O$ is pumped into a second tank (6), and has $H_2$ bubbled through it to purge the $CO_2$. From there, the $H_2O$ is pumped into the electrolyzer (8), which separates a portion of it into $O_2$ and $H_2$. The $H_2$ re-enters the gas loop via (12), while the remaining $H_2O$, along with the $O_2$, goes into tank (7), where the mixture is cooled and separated. The $H_2O$ returns to the electrolyzer, while the $O_2$ leaves the system through (14).

In addition to its normal operating mode, the system may operate without the electrolyzer and water pumps. In this mode, the $H_2$ for the reaction is supplied by a supply line (15) paralleling the $CO_2$ supply line. This option is not feasible for operation on Mars, but has proven useful for testing the physical system while under development.

The RWGS is an interconnected nonlinear system where the various components influence each other in complicated and sometimes unexpected ways. For example, during runs without the electrolyzer, it is necessary to empty the water tank (5) periodically, to prevent water from accumulating and eventually overflowing the tank. This causes the gases in the tank to expand, and thus creates a significant and sudden pressure drop, which affects the flow throughout the whole system. This phenomenon is demonstrated in Fig. 2(b), taken from [Whitlow, 2001]. The graph shows the flow through the CO vent (13) as it evolves over time — the spikes correspond to emptying the water tank.

A challenging property of the RWGS is that phenomena in the system manifest themselves over at least three different time scales. Pressure waves in the RWGS propagate essentially instantaneously (at the speed of sound). Gases flow around the gas loop on the order of seconds. Finally, gas compositions in the gas loop take on the order of hours to reach a steady state. Meanwhile, the sensors collect data at a sampling rate of one second.

An additional challenge of the RWGS is its sensitivity and unidentifiability, i.e., parts of the system state are very sensitive to input paramaters and are not directly measured. For example, the $H_2$ and $CO_2$ compositions in the gas loop cannot be practically measured. However, the balance between these compositions is almost neutrally stable, thus a small shift in the input conditions or the membrane behavior will cause the balance to gradually drift to a significantly different value.

As in any real system, the RWGS sensors do not record the underlying state exactly. In addition to some important quantities, such as the gas compositions, which are not measured at all, the existing sensors are noisy and biased. The noise level of the sensors depends on many factors and can change over time. An example is shown in Fig. 2(c), where the difference in the readings of the pressure sensors $P_3$ and $P_4$ (both located at (2) in Fig. 2(a)) is plotted over time. The main reason for the noise in time steps 0–800 is the physical proximity of the sensors to the compressor that sends pressure waves throughout the system. Since the sensors are not synchronized with the compressor, they take measurements at various phases of the pressure waves

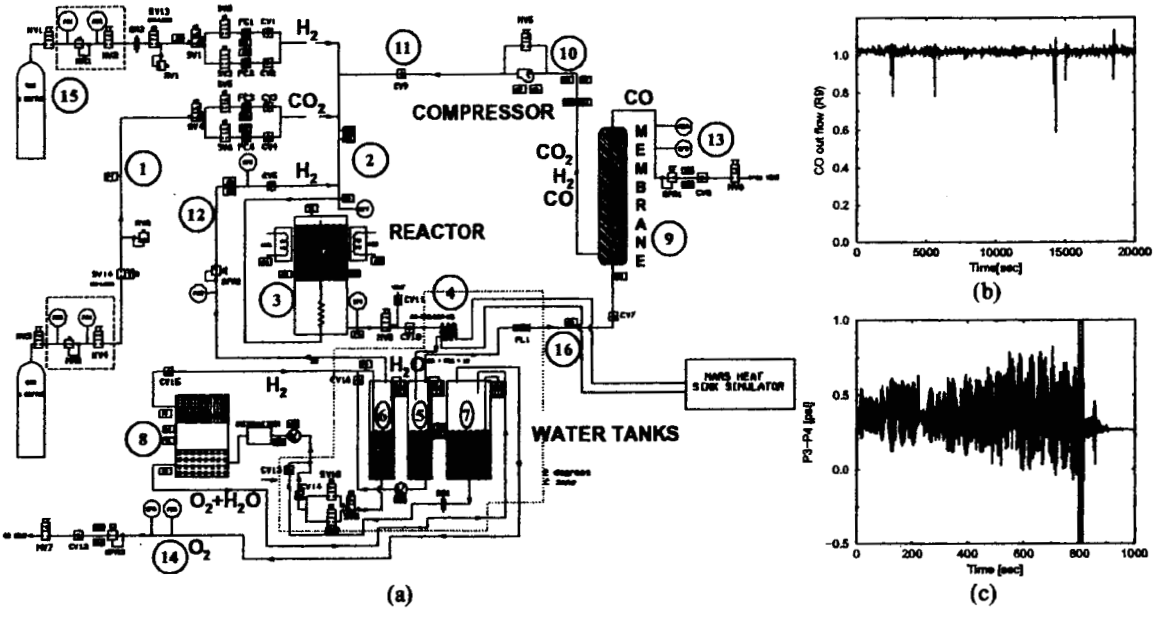



Figure 2: (a) The RWGS Schematic. (b) Effects of emptying a water tank. (c) Pressure difference between $P_3$ and $P_4$.

and thus measure significantly different values. After 796 seconds the compressor shuts down and the noise level decreases dramatically. [2] More interestingly, we note that the two sensors are placed very close together and thus the average difference should be zero. However, as the plot demonstrates, this is not the case, indicating that the sensors are not well calibrated and some bias is present. Furthermore this bias depends on the system state, as shown by the change in the average difference when the compressor shuts off.

## 4 Modeling the RWGS

We model the RWGS using a hybrid DBN, as described in Section 2. The 2TBN has 293 nodes, 227 of which are continuous. Currently the discrete variables in the model are all known and correspond to computer-controlled switches and sensor faults. The continuous variables in our model capture the continuous-valued elements of our system (e.g., pressure at various points in the system, flow rates, temperatures, gas composition, etc.). Of the 227 continuous nodes, 43 represent the time $t$ belief state $\boldsymbol{X}$ and 184 represent the variables $\boldsymbol{X}'$ at time $t+1$. Of the latter, 43 variables are belief state variables for $t+1$, 72 variables are *encapsulated* variables, as discussed in Section 5.4, and the rest are either sensor variables or transient variables.

When constructing the model, we used four techniques for parameter estimation. Some of the parameters were known physical constants or system properties. Of the empirical parameters, many came from physical models. The others (specifically, some parameters for the compressor, the separation membrane and the overall system pressure changes) were determined using common equations that model the particular system behavior. All the variables in these equations were directly observed in the data, and thus we could use least-squares techniques to find the best fit for the parameters. The remaining parameters were estimated using prior knowledge of the domain.

### 4.1 Sensor Modeling

As discussed in Section 3, one of the challenges we address in modeling the RWGS is dealing with noisy and biased sensors. We deal with noisy sensors in the obvious way: by increasing the variance of the predicted measurement values to match the noise level in the data.

Sensor biases present a more interesting modeling problem. The biases are not easily modeled using a simple parameter since they are unknown and can drift over time. Instead, we address the problem by adding hidden variables to the belief state that model the different biases of the sensors. Biases start with zero mean and a reasonably large variance and persist over time, i.e., $\text{Bias}^{t+1} = \text{Bias}^{t} + V^{t}$, where $V^{t}$ represents white noise with a relatively small variance, allowing for some amount of drift to occur over time.

This idea works quite well, but it tends to overfit the data: By letting the bias account for every discrepancy between the model predictions and the actual sensor measurements, the tracking algorithm might settle in an incorrect steady state. To fix the problem we must make sure that the model biases reflect true sensor biases — biases should

[2] The sensor's noise is literally noise that can be heard — the pressure waves are the sound waves generated by the compressor.

be kept as small as possible and allowed to grow only if there is a real reason for that. We implement this idea by introducing a contraction factor $\gamma < 1$ (empirically set to be 0.97) into the bias formula: $\text{Bias}^{t+1} = \gamma \cdot \text{Bias}^t + V$. Thus, biases tend to go to zero unless doing so introduces a systematic discrepancy with the predicted system state.

### 4.2 Sensitivity and Unidentifiability

Recall that the equations governing the $H_2/CO_2$ balance in the gas loop are sensitive to slight variations in the physical parameters. Thus even using the most exact form of these equations in the model will result in (at least) the same level of sensitivity — both to variations in the physical parameters, and inherent errors in the parameters. Moreover, the model value is also sensitive to model effects such as calculation errors and sensor errors that do not affect the real value. We therefore use equations for the $H_2/CO_2$ balance that contain an intentionally non-physical component—a *stabilizing* term—that reduces the sensitivity. This term drives the balance to a pre-determined point, which in this case is our expected value for the balance. The magnitude of this term is manually adjusted to provide an optimum tradeoff between physical accuracy and model stability.

### 4.3 Differing Time Scales

As described in Section 3, we must deal with differing time scales in modeling the RWGS. The naive solution to this problem is to model the DBN at a very fine time granularity. However, it is completely impractical to model the behavior of the pressure waves using a discretized-time model. To do so would require time steps three orders of magnitude smaller than the time between measurements, which is a significant waste of resources. Furthermore, it would require a much more complete description of the system than is practical, and tracking the slowly-evolving aspects of the system with a step size many orders of magnitude below their time scale would allow substantial errors to build up.

Thus, we approximate the pressure waves as occurring instantaneously and instead of modeling their transient behavior, we model the quasi-steady-state results at each time step after they have reached an equilibrium. The equations in this case are substantially simpler, and require far fewer empirical constants. The difficulty, however, is that these equations must be solved simultaneously; a change in any part of the system will affect all of the other parts. These equations include both the compressor equation and an approximation to the membrane equations developed in [Whitlow, 2001]; thus, they are fairly large and nonlinear, and no direct simultaneous solution form exists. Instead, we use these equations to create a new equation that converges to a fixed point solution.

We must insert this fixed-point equation into a (nonlinear) CPD to use it in our DBN model of the RWGS. The equation solves for the five model variables $Z$ that account for the flows and pressure of the gas loop. In order to solve for all five variables, their eight parents must also be present in the CPD. Hence, we have a vector CPD for $Z$ whose definition is essentially procedural: given a value of the eight parents it executes an iterative fixed-point computation until convergence, and outputs the values $Z$.

## 5 Tracking in Nonlinear Systems

In this section, we address the problem of inference, focusing on tracking in complex nonlinear systems, such as the RWGS. In these models, the probabilistic dependencies, including sensors, can be either linear or nonlinear functions with Gaussian noise. We restrict our attention to the task of tracking the continuous state, assuming all the discrete values are known. Note that although the results in this section are presented in terms of dynamical systems, the analysis also applies to probabilistic inference in *static* nonlinear Bayes nets.

### 5.1 Exploiting DBN Structure

Recall the setup from Section 2: We have a Gaussian belief state $\text{Bel}(\boldsymbol{X})$ where $\boldsymbol{X} \in \mathbb{R}^n$ and a 2TBN representing $P(\boldsymbol{X}' \mid \boldsymbol{X})$ as a deterministic function $\boldsymbol{X}' = f(\boldsymbol{X})$. Our goal is to find an approximation of $P(\boldsymbol{X}')$ as a multivariate Gaussian. The classical approach, used in the EKF and the UF, is to find the entire distribution $P(\boldsymbol{X}')$ directly by treating $f$ as a function from $\mathbb{R}^n$ to $\mathbb{R}^m$. An alternative approach is to decompose $f$ by defining $X_i' = f_i(\boldsymbol{Y}_i)$ for $i = 1, \ldots, m$, where $\boldsymbol{Y}_i = \text{Par}(X_i')$. In most practical cases the $f_i$'s have a lower dimension than $f$; as we shall see, this reduction in the dimension lets us approximate the resulting distribution more accurately and efficiently.

As discussed in Section 2, the first step in the belief state propagation process is to compute a multivariate Gaussian over $\{\boldsymbol{X}, \boldsymbol{X}'\}$. We begin with our Gaussian $\text{Bel}(\boldsymbol{X})$, and add the variables from $\boldsymbol{X}'$ one at a time, using the procedure described in Section 5.2. The key insight is that, as $X_i'$ is conditionally independent of $\{\boldsymbol{X} - \boldsymbol{Y}_i, X_1', \ldots, X_{i-1}'\}$ given $\boldsymbol{Y}_i$, it suffices to approximate the Gaussian $P(\boldsymbol{Y}_i, X_i')$. We can then compute $P(\boldsymbol{X}, X_1', \ldots, X_i') = P(\boldsymbol{X}, X_1', \ldots, X_{i-1}')P(X_i' \mid \boldsymbol{Y}_i)$, which, for Gaussians, can be accomplished using simple linear algebra operations.

A more difficult case arises when the DBN contains not only inter-temporal edges from $\boldsymbol{X}$ to $\boldsymbol{X}'$, but also intra-temporal edges between $\boldsymbol{X}'$ variables. In this case we sort the variables $X_i'$ in topological order, and gradually build up the joint distribution $P(\boldsymbol{X}, X_1', \ldots, X_i')$. The topological order ensures that when we need to compute $P(\boldsymbol{Y}_i, X_i')$, we have already computed a Gaussian over $\boldsymbol{Y}_i \subseteq \boldsymbol{X} \cup \{X_1', \ldots, X_{i-1}'\}$. This approach, however, may introduce some new inaccuracies, because we now also use a Gaussian approximation for the distribution of the relevant variables from $\{X_1', \ldots, X_{i-1}'\}$.

Even in cases where we introduce extra inaccuracies, this method is often superior to the UF. The reason is that, by reducing the dimension of the functions involved, we

can use more accurate techniques to approximate the first two moments of the variables in $X'$ with the same computational resources. In general, there is a tradeoff between the superior precision we achieve for each variable and the potential for extra inaccuracies we introduce. The extra inaccuracies depend on the quality of our Gaussian approximation for $P(X, X'_1, \ldots, X'_{i-1})$, and on the extent of the nonlinearity of the dependencies within $X'$. If the dependence of $X'_i$ on $\{X'_1, \ldots, X'_{i-1}\}$ is linear, then there are no extra errors introduced: In this case the first two moments of $X'_i$ are only influenced by the first two moments of $\{X'_1, \ldots, X'_{i-1}\}$ which can be captured correctly by our Gaussian approximation. *It is somewhat reassuring that the* better our approximation of $P(X')$ as a Gaussian is, the less significant the extra errors we introduce are, as the entire framework is based on the assumption that $P(X')$ can be well approximated by a Gaussian.

### 5.2 Numerical Integration

We now turn our attention to the task of approximating $P(Y_i, X'_i)$ as a multivariate Gaussian. To simplify our notation, let $X$ be a variable which is a nonlinear function of its parents $Y = Y_1, \ldots, Y_d$, i.e., $X = f(Y)$, but the ensuing discussion also holds for the vector case of $X = f(Y)$. We assume that $P(Y)$ is a known multivariate Gaussian, and the goal is to find a Gaussian approximation for $P(Y, X)$. It suffices to compute the first two moments:

$$E[X] = \int P(Y)f(Y)dY \quad (1)$$

$$E[X^2] = \int P(Y)f^2(Y)dY \quad (2)$$

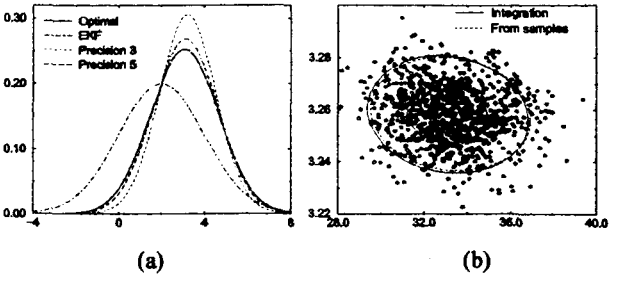
$$E[XY_j] = \int P(Y)f(Y)Y_j dY \quad (3)$$

Note that the integrals only involve the direct parents of $X$, significantly reducing their dimension. We can effectively compute these integrals using a version of the Gaussian Quadrature method called the *Exact Monomial* rules [Davis and Rabinovitz, 1984]. Generally speaking, Gaussian Quadrature approximates integrals using a formula of the form:

$$\int W(x)f(x)dx \approx \sum_{j=1}^{N} w_j f(x_j)$$

where $W(x)$ is a known function (a Gaussian in our case). The points $x_j$ and weights $w_j$ are carefully chosen to ensure that this approximation is exact for any polynomial $f$ whose degree is at most $p$. The degree $p$ is called the *precision* of the approximation.

Finding a set of points with a minimal size $N$ for some precision $p$ is not a trivial task. In the simple form of Gaussian Quadrature, we choose points in one dimension and use them to create a grid of points in $\mathbb{R}^d$ with the obvious disadvantage that $N$ grows exponentially with $d$. Fortunately, we can do better. In [McNamee and Stenger, 1967] a general method is presented for $N = O\left(\frac{(2d)^k}{k!}\right)$ and precision $p = 2k + 1$ ($d$ is the dimension of the integral, in our case $|Y|$). In particular, rules are presented for $2d + 1$ points with precision 3, $2d^2 + 1$ points with precision 5 and $\frac{4}{3}d^3 + \frac{8}{3}d + 1$ points with precision 7. The precision 3 rule is exactly the rule used for the Unscented Filter: It has exactly the same $2d + 1$ points and weights.




(a) (b)

Figure 3: (a) Density estimates for $N = \sqrt{Y_1^2 + Y_2^2}$. (b) Random samples from the RWGS network for the flow at point (16) and the pressure at point (2), and Gaussian estimates for the distribution.

This view of the Unscented Filter has immediate practical consequences: we can trade off between the accuracy of the computation and its computational requirements. For example, if we are interested in a more precise filter than the Unscented Filter and are willing to evaluate the function at $O(d^2)$ points then we can use the exact monomial rule of precision 5. Depending on the function, this may represent a significant gain in accuracy.

As a simple example we consider the nonlinear function $N = \sqrt{Y_1^2 + Y_2^2}$ where $P(Y_1) = \mathcal{N}(2, 4)$ and $P(Y_2 \mid Y_1) = \mathcal{N}(0.5Y_1 - 1, 3)$ (note that both $Y_1$ and $Y_2$ have the same variance 4). Fig. 3(a) shows various estimates for the probability of $N$. The optimal estimate is the best Gaussian approximation for the distribution of $N$ computed using a very exact numerical integration rule. We can see that the exact monomial rules of precisions 3 and 5 provide a much better estimate than EKF, where the precision 5 rule leads to a more accurate estimate than the precision 3 rule.

### 5.3 Inaccuracies in the Approximation

Unfortunately, approximating $P(Y, N)$ using numerical integration can lead to covariance matrices that are not semi-positive definite, and hence illegal. One simple approach to this problem is to use a more accurate integration rule, although the problem may persist. An alternative is to find the "closest" positive definite covariance matrix. We cast this problem as a convex optimization problem following [Boyd and Vandenberghe, 2003].

Consider once again the problem of approximating $P(Y, N)$ as a multivariate Gaussian, where $N$ is a nonlinear function of its parents $Y$, i.e., $N = f(Y)$, and $Y \sim \mathcal{N}(\mu_Y, \Sigma_{YY})$. Let $\Sigma$ denote the estimated covari-

ance matrix for $P(\boldsymbol{Y}, X)$:

$$\Sigma = \begin{pmatrix} \Sigma_{\boldsymbol{Y}\boldsymbol{Y}} & \bar{u} \\ \bar{u}^T & \bar{v} \end{pmatrix}$$

If $\bar{u}$ and $\bar{v}$ lead to a matrix $\Sigma$ that is not positive definite, then we need to find the closest $u$ and $v$ to $\bar{u}$ and $\bar{v}$, such that $\Sigma$ is positive definite. Given that $\Sigma_{\boldsymbol{Y}\boldsymbol{Y}}$ is already positive definite, $\Sigma$ is positive definite iff $v - u^T \Sigma_{\boldsymbol{Y}\boldsymbol{Y}}^{-1} u > 0$. Thus, we can formalize our problem as follows:

$$\text{Minimize} \quad \| u - \bar{u} \|^2 + (v - \bar{v})^2 \qquad (4)$$

$$\text{Subject to} \quad u^T \Sigma_{\boldsymbol{Y}\boldsymbol{Y}}^{-1} u - v + \epsilon \leq 0 \qquad (5)$$

where $\epsilon$ is some small positive number. Since both Eq. 4 and Eq. 5 are convex we can solve this problem by forming the Lagrangian and solving the dual problem. We set the partial derivatives of $u$ and $v$ to zero and plug the result into Eq. 5. We get an equation over the Lagrangian multiplier which can be solved easily as it involves a monotonic function. We omit details for lack of space.

Our analysis treats the elements in $u$ and $v$ directly, but in fact these elements are not independent since $u_i = E[Y_i X] - \mu_{Y_i} E[X]$ and $v = E[X^2] - E[X]^2$. It is desirable to use this relation in Eq. 4 and Eq. 5 and represent the dependency between the various elements (e.g., a change in $E[X]$ may fix many of the problems simultaneously). Unfortunately, because of the term $E[X]^2$ the problem is no longer convex. Nonetheless, we can approximate the problem as convex (e.g., by replacing $E[X]^2$ by the best current estimate), solve it and iterate. Again, we defer details to an extended version of this paper.

### 5.4 Encapsulated Variables

Just as we can use the DBN structure to decompose the dependency between $\boldsymbol{X}'$ and $\boldsymbol{X}$, in many cases we can further decompose the dependency $X = f(\boldsymbol{Y})$. For example, assume that $f(\boldsymbol{Y}) = g(g_1(\boldsymbol{Y}_1), g_2(\boldsymbol{Y}_2))$, where $\boldsymbol{Y}_1, \boldsymbol{Y}_2 \subseteq \boldsymbol{Y}$.[3] Instead of directly evaluating the Gaussian over $\{\boldsymbol{Y}, X\}$ we can define two extra variables: $T_1 = g_1(\boldsymbol{Y}_1)$ and $T_2 = g_2(\boldsymbol{Y}_2)$. We first approximate $P(\boldsymbol{Y}_1, T_1)$ as a Gaussian and use it to find a Gaussian over $\{\boldsymbol{Y}, T_1\}$. Next we approximate $P(\boldsymbol{Y}_2, T_2)$ as a Gaussian and from it $P(\boldsymbol{Y}, T_1, T_2)$. Finally, we approximate $P(T_1, T_2, X)$ as a Gaussian and use it to find the Gaussian approximation for $P(\boldsymbol{Y}, T_1, T_2, X)$. The same accuracy tradeoffs that were discussed in the context of $\boldsymbol{X}' = f(\boldsymbol{X})$ apply here: by reducing the dimension of the integrals we can solve each one more accurately, but may introduce further errors if the interaction between the extra variables is nonlinear.

[3] E.g., flow sensors give different results depending on the gas type. Assuming we have random variables representing the total flow and the compositions of the different gases in it, $g_1$ and $g_2$ may each be a product of one of the gas compositions and the flow, thus representing the net flow of a certain gas. The function $g$ would be a weighted sum of these flows where the weights correspond to the sensor's response for the different gases.

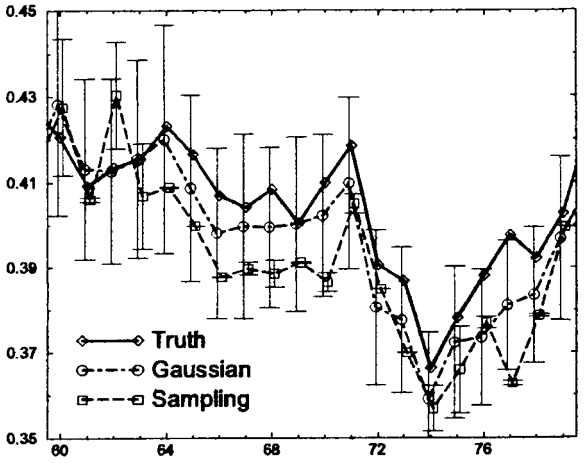



Figure 4: Comparison with particle filtering on simulated data, showing the means and error bars of two standard deviations for our algorithm and particle filtering. The $X$ axis represents time, and the $Y$ axis the percentage of $H_2$ in the flow at point (16). To increase readability, we shift the estimates generated by our algorithm by 0.1 to the left and those generated by particle filtering by 0.1 to the right.

In principle, one could add $T_1$ and $T_2$ to the DBN and treat them as regular variables. However, doing so makes these variables part of $\boldsymbol{X}'$, and thereby increases the algorithm's space complexity, which is $O(|\boldsymbol{X}'|^2)$ (for representing the covariance matrix of $P(\boldsymbol{X}')$). It is better to treat the extra variables as local variables *encapsulated* within the CPD and unknown to the rest of the network. After computing the Gaussian approximation for the CPD variables, we simply marginalize over the encapsulated ones. This approach is similar to the local computations in an OOBN model [Koller and Pfeffer, 1997], where some of the CPD variables are encapsulated within the CPD.

## 6 Experimental Results

In this section we present results from a set of experiments that test the efficacy and robustness of our model and tracking algorithm. Our computational model of the RWGS contains all of the components needed to monitor the full operation of the physical system, although data provided to date by KSC is for the reduced-operation mode with only the gas loop component operational. Our experiments were run on a Pentium III 700MHz.

We tested our algorithm with both real data and simulated data that was generated from our model. Although running with real data is the real test for our approach, running with simulated data is also of interest. The reason is

that there are two sources of errors when using real data: model inaccuracies and errors induced by the algorithm. When using simulated data, only errors of the second type are present and we can better test the performance of the algorithm.

### 6.1 Results on Simulated Data

We first tested whether the belief state could be well approximated as a Gaussian and whether our particular approximation was a good one. To do so, we generated a set of samples from the model. We did not introduce any evidence so the samples were indeed sampled from the correct joint distribution. In Fig. 3(b) we show the results for two particular variables: the flow at point (16) and the pressure at point (2) (these variables were chosen because of their dependency on the non-linear CPD of the membrane; other variables produced similar results). The samples appear to be drawn from a distribution that is either a Gaussian or close to one. Furthermore, our estimate for the joint distribution (depicted by the contours for one and two standard deviations) is very close to the Gaussian that was estimated directly from the samples. Thus, it is reasonable to expect that our techniques will lead to good approximations of the belief state.

Next, we generated a trajectory of 500 time steps from our model and tested our algorithm on it. We compared our results with the particle filtering algorithm [Gordon *et al.*, 1993], which approximates the belief state as a set of weighted samples where the weights of the samples correspond to the likelihood of the evidence given the sample. Our algorithm took 20ms per time step, which included computing the Gaussian approximation to the belief state, with numerical integration when necessary, and conditioning on the evidence. In comparison, generating a sample using particle filtering took 1.5ms. Thus, one step of our algorithm took as much time as generating 13 samples. However, with just 13 samples particle filtering performed extremely poorly and therefore in our experiments we used 10,000 samples at every time step, giving particle filtering a somewhat unfair advantage.

Fig. 4 shows the estimates for the percentage of $H_2$ in the flow at point (16) that were computed by our algorithm and by particle filtering, as well as the actual value (known from the simulated data). We report the results on this particular variable because the gas compositions are not measured by any sensors and are therefore a potential challenge to our algorithm. The error bars represent the uncertainty of the estimates as plus and minus two standard deviations (for particle filtering we computed the standard deviation induced by the weighted samples).

Although under our setup particle filtering was slower than our algorithm by a factor of 750, as Fig. 4 demonstrates, the estimates of particle filtering are not as good as the estimates of our algorithm. Over the entire sequence the average error of our algorithm was 0.009 while the average error of particle filtering was 0.013. Nevertheless, the more dramatic difference is in the estimates of the variance. Often, the estimated variance for particle filtering is extremely small, even when the estimated value is not very accurate (e.g., time steps 72 and 73). In fact, over the entire sequence, according to the estimated distribution of our algorithm, the correct value of the $H_2$ composition was within two standard deviations 96% of the time (this is consistent with the fact that the probability mass within two standard deviations from a Gaussian mean is 95%). In comparison, for particle filtering, the true value was within two estimated standard deviations only 20% of the time. The difference was even more apparent when we computed the average log-likelihood of the true value, given the two possible estimates. For our algorithm the average log-likelihood was 3.1 while for particle filtering it was only $-5.59 \cdot 10^{11}$.

The reason for this problem is the relatively high dimension of the evidence which leads to a very high variance for the weights of the samples. Although we generated 10,000 samples at each time step only a very small number of them had a significant effect on the estimate. Over the entire sequence, in 65% of the time steps one sample accounted for more than 0.5 of the total probability mass, in 27% one sample accounted for more than 0.9 of the mass, and in 15% one sample accounted for more than 0.99. Obviously in cases where one sample completely dominates the rest, the estimates of particle filtering are not very reliable and in particular the variance estimates can be extremely small and misleading.

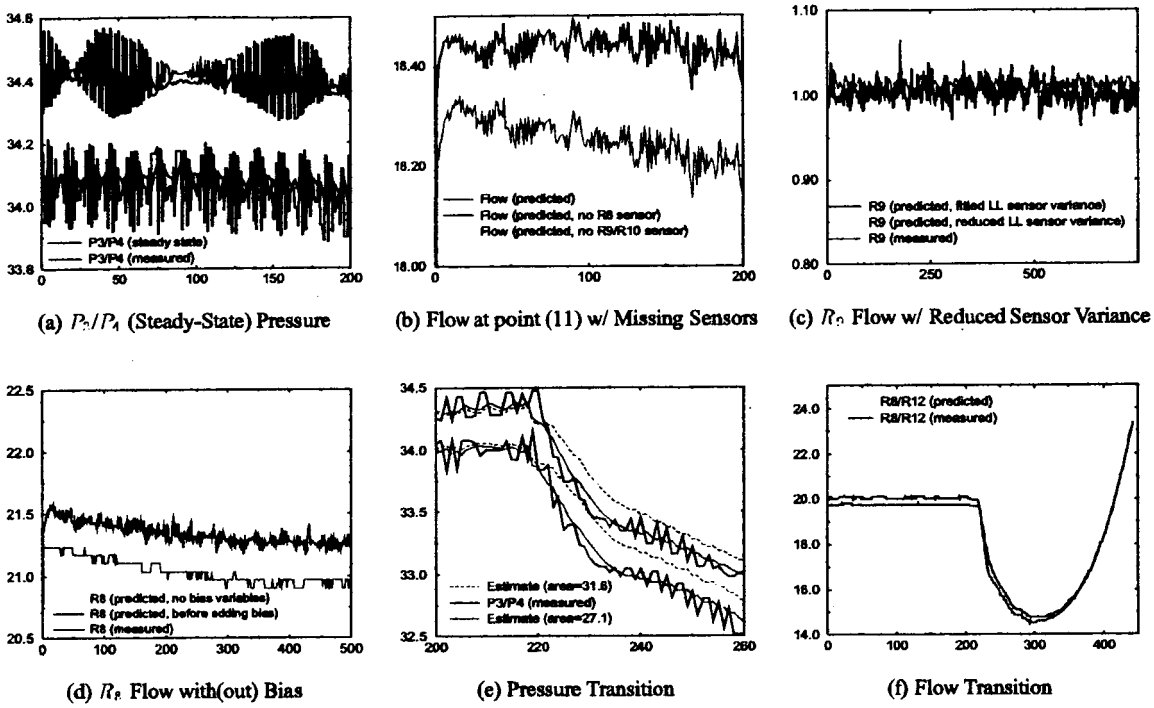
Thus, not only is our algorithm faster than particle filtering with 10,000 samples by a factor of 750, its estimates are much more reliable.

### 6.2 Results on Real Data

We next ran a set of experiments on real data. Our data set consisted of a long sequence of 13,875 time steps, most of it collected while the system was running in steady state. We divided our data into a training set, used to estimate and tune the model parameters, and a test set on which we report our results.

We conducted a variety of experiments in which we compared model predictions with the actual measurements recorded by the system under various scenarios: steady state and non-steady state, removing sensors, and modifying the sensor models. In order to make the comparison informative, the model predictions for values at time $t+1$ as reported in this section are not adjusted with evidence at time $t+1$, i.e., they are the predictions based on evidence from times $0, 1, \ldots, t$.

Our first experiment, shown in Fig. 5(a), illustrates the efficacy of our tracking algorithm during steady-state operation of the system. In particular, the graph illustrates the predicted (thick lines) and measured (thin lines) pressures, $P_3$ and $P_4$ at point (2) in Fig. 2(a). Observe that the predicted value for $P_3$ appears to be consistently lower than the measurement. This is the result of the model's bias weighting, $\gamma = .97$, discussed in Section 4.1, which




(a) $P_3/P_4$ (Steady-State) Pressure

(b) Flow at point (11) w/ Missing Sensors

(c) $R_9$ Flow w/ Reduced Sensor Variance

(d) $R_8$ Flow with(out) Bias

(e) Pressure Transition

(f) Flow Transition

Figure 5: Experimental Results Tracking the RWGS. The $X$ axis represents time, and the $Y$ axis the value of the appropriate quantity.

tends to pull the estimates slightly away from the measured value. While, in this case, it produces a slightly poorer result, overall, the bias weighting technique does less data overfitting and works better in non-steady state sequences.

Next we experimented with "removing" sensors from the system. (This is easily achieved by ignoring selected sensor evidence when running the tracking algorithm.) Sensor removal can be used to evaluate the robustness of the algorithm as well as to determine the importance of a sensor for monitoring the system. In Fig. 5(b), we show the flow of gas from the compressor at point (11). The two overlaid lines are our estimates of this flow value — one with all of the sensors, and the other with sensors $R_9$ and $R_{10}$ (located at (13)) removed. In contrast, when flow sensor $R_8$ (located at (16)) is removed, the predicted flow rate quickly strays. These results indicate that, at least for this sequence, $R_8$ is a more valuable sensor than $R_9$ and $R_{10}$.

We also tested the effects of changing the liquid level (LL) sensor noise parameter [4] on our prediction of the gas flow $R_9$ at (13). Recall from Section 4.1 that to correctly model a sensor we introduced both some Gaussian noise on the sensor and a hidden bias variable. We tried both a variance value of 0.01, which we estimated using "reasonable" prior knowledge, and a variance value of 4 which was fit to the data. Fig. 5(c) shows the effect of the variance of the LL sensor for the water tank at (5). With the fitted variance, the algorithm tracked quite well. In contrast, with the smaller variance, the performance was poor and erratic, following the fluctuations in the LL measurements.

The utility of the bias variables is shown in Fig. 5(d). The upper line is a prediction of the flow rate, made using a version of the model that contained no bias variables for the flow sensors at (10), (13) and (16). The middle line corresponds to the model with the bias variables present, but shows the prediction for the true (unbiased) flow (i.e., the sensor prediction minus the bias). When we explicitly modeled the sensor bias, our (unbiased) predictions of the true system state better matched the measurements, an indication of a better estimate of the system state.

Finally, we tested the ability of the model to track non-steady-state behavior — in particular, the behavior of the system when the $CO_2$ supply is turned off during the shutdown process. Unfortunately, we only had one data set containing this transition, and thus we expect our parameters are still not tuned optimally. In addition, having only one such transition in our data, we report results on the same data that was used for training.

Fig. 5(e) shows a comparison between the predicted and

[4] The liquid level sensor is very noisy, as splashing and bubbling from the dissolved $CO_2$ and from drops splashing from the condenser hit the sensor rod and create considerable noise in the sensor reading.

measured output from pressure sensors $P_3$ and $P_4$, for two versions of the model. The first set of predictions, shown in solid lines, was calculated using our best estimates of the empirical parameters, including the membrane area (calculated from other parts of the data set) of 27.1. The second set of predictions, shown in dashed lines, was calculated using an earlier estimate of the membrane area of 31.6. While in the steady-state prior to timestep 220, the two predictions are equivalent as the differences were absorbed into the bias errors, in the transient part, the model with inaccurate parameters underpredicts the initial drop in pressure, and retains this error throughout the rest of the sequence.

Fig. 5(f) presents the predictions of the correct model for the flows at $R_8$ (16) and $R_{12}$ (10), over a longer period of time. Initially, when the $CO_2$ supply was cut off, the flows dropped; however, gradually the CO and $CO_2$ in the system were vented and the only remaining gas was $H_2$. As the membrane presented less resistance to $H_2$ the flow rates started to go up. The model tracked this complex behavior surprisingly well.

## 7 Conclusions and Future Work

In this paper we address the problem of monitoring a large complex physical system — NASA's Reverse Water Gas Shift system — perhaps the largest and most complex hybrid DBN developed to date. This paper makes contributions both to the modeling and the monitoring of complex nonlinear systems. On the modeling side, we have shown how to model physical systems whose effects manifest themselves at dramatically different time scales, and that involve biased sensors, where the bias is state dependent and varies over time. On the monitoring side, we have presented a general framework for approximating nonlinear behavior using integration methods that extend the Unscented Filter, improving the accuracy of the approximation with minimal additional computation. Experimental results indicate that this approach is much faster and more reliable than particle filtering. More generally, we have demonstrated the feasibility of hybrid DBNs for monitoring a complex real-world physical system such as the RWGS using real data.

There are several interesting directions for future work. The tracking algorithms presented in this paper assume a known mode of operation, i.e., all the discrete variables are observed. Our long-term goal is to diagnose the RWGS when components fail. In order to track both the discrete and continuous state, we intend to combine the results presented in this paper with algorithms that handle hidden discrete events such as *Rao-Blackwellized Particle Filtering (RBPF)* [Doucet *et al.*, 2000] or the algorithms presented in [Lerner and Parr, 2001; Lerner *et al.*, 2000]. The speed of our algorithm (taking just 20ms to generate a Gaussian over all the state variables) is a promising indication that we can use these techniques for real-time fault diagnosis.

**Acknowledgements**

We are very grateful to Charlie Goodrich and the rest of the RWGS team at the Kennedy Space Center — Bill Larson, Clyde Parrish, Jon Whitlow, Curtis Ihlefeld, and Dan Keenan — for their tremendous help and support. We also thank Dan Clancy, Ronald Parr, and Stephen Boyd for useful suggestions and discussions. This research was supported by ONR Young Investigator (PECASE) under grant number N00014-99-1-0464, by ONR under the MURI program "Decision Making under Uncertainty", grant number N00014-00-1-0637, and by NASA under grant number NAG2-1337.

## References

[Bar-Shalom *et al.*, 2001] Y. Bar-Shalom, X. R Li, and T. Kirubarajan. *Estimation with Application to Tracking and Navigation*. John Wiley & Sons, 2001.

[Boyd and Vandenberghe, 2003] S. Boyd and L. Vandenberghe. *Convex Optimization*. 2003. To appear.

[Davis and Rabinovitz, 1984] P. J. Davis and P. Rabinovitz. *Methods of Numerical Integration*. Academic Press, 1984.

[Dean and Kanazawa, 1989] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[Doucet *et al.*, 2000] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[Goodrich, 2002] C. Goodrich. Reverse water gas shift system presentation. Stanford University, April 2002.

[Gordon *et al.*, 1993] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140(2):107–113, April 1993.

[Julier and Uhlmann, 1997] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[Kalman, 1960] R.E. Kalman. A new approach to linear filtering and prediction problems. *J. of Basic Engineering*, 82:34–45, 1960.

[Koller and Pfeffer, 1997] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proc. UAI*, pages 302–313, 1997.

[Larson and Goodrich, 2000] W. Larson and C. Goodrich. Intelligent systems software for human mars missions. In *2000 International Aeronautical Foundation Congress*, 2000.

[Lerner and Parr, 2001] U. Lerner and R. Parr. Inference in hybrid networks: Theoretical limits and practical algorithms. In *Proc. UAI*, pages 310–318, 2001.

[Lerner *et al.*, 2000] U. Lerner, R. Parr, D. Koller, and G. Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *Proc. AAAI*, pages 531–537, 2000.

[McNamee and Stenger, 1967] J. McNamee and F. Stenger. Construction of fully symmetric numerical integration formulas. *Numerische Mathematick*, 10:327–344, 1967.

[Whitlow, 2001] J. E. Whitlow. Operation, modeling and analysis of the reverse water gas shift process. In *NASA CR-2001-(In Press)*, 2001.

# What Sensing Tells Us: Towards A Formal Theory of Testing for Dynamical Systems

**Sheila A. McIlraith**
Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305-9020
sam@ksl.stanford.edu

**Richard Scherl**
Department of Computer Science
New Jersey Institute of Technology
University Heights
Newark, NJ 07102-1982
scherl@cis.njit.edu

**Abstract**

Just as actions can have indirect effects on the state of the world, so too can sensing actions have indirect effects on an agent's state of knowledge. In this paper, we investigate "what sensing actions tell us", i.e., what an agent comes to know indirectly from the outcome of a sensing action, given knowledge of its actions and state constraints that hold in the world. To this end, we propose a formalization of the notion of testing within a dialect of the situation calculus that includes knowledge and sensing actions. Realizing this formalization requires addressing the ramification problem for sensing actions. We formalize simple tests as sensing actions. Complex tests are expressed in the logic programming language Golog. We examine what it means to perform a test, and how the outcome of a test affects an agent's state of knowledge. Finally, we propose automated reasoning techniques for test generation and complex-test verification, under certain restrictions. The work presented in this paper is relevant to a number of application domains including diagnostic problem solving, natural language understanding, plan recognition, and active vision.

## Introduction

Agents equipped with perceptual capabilities must operate in a world that is only partially observable. To determine properties of the world that are not directly observable, an agent must use its knowledge of the relationship between objects in the world, and its limited perceptual capabilities to infer such unobservable properties. For example, if an agent performs a sense action and observes that there is steam coming out of an electric kettle, then the direct effect of that sensing action is that the agent *knows* there is steam coming out of the kettle. With appropriate knowledge of the functioning of kettles, the agent should also know that the electrical outlet has power, that the kettle is functioning, and that there is hot liquid inside the kettle – all as *indirect* effects of the sensing action. Similarly, if the agent wishes to know whether there is power at an electrical outlet, but cannot directly sense this property of the world, the agent may potentially acquire this knowledge by attempting to boil water in a kettle plugged into this outlet.



Such a sequence of actions constitutes a *test*. If steam is observed, then the agent knows that there is power at the outlet; however if steam is not observed, the agent may or may not know that there is no power at the electrical outlet. The knowledge the agent acquires from the test will depend on whether the agent knows that the kettle is functioning. Thus, this particular test is only *guaranteed* to provide knowledge about the existence of power at the electrical outlet under one test outcome.

While researchers have extended theories of action to include the notion of *sensing* or *knowledge-producing actions* (e.g., (Scherl & Levesque 1993; Baral & Tran 1998; Golden & Weld 1996; Funge 1998)) and have characterized the effect of sensing actions on an agent's state of knowledge, and even how to plan (e.g., (Stone 1998; Golden & Weld 1996)) and to project (e.g., (De Giacomo & Levesque 1999b)) in certain cases, with sensing actions, they have not addressed the problem of how to reason in a partially observable environment[1]. More generally, they have not examined the problem of how sensing actions can be coupled with knowledge of the relationship between objects in the world to gain further knowledge, and how both sensing actions, and world-altering actions change an agents state of knowledge in the presence of such world knowledge. Further, they have not examined the problem of how to select sensing actions to acquire knowledge of some property of the world that is not directly observable. Perhaps the closest research is that of (Shanahan 1996b; 1996a) who investigates the assimilation of sensing results for a mobile robot in a framework based on the event calculus, (McIlraith 1997) who assimilates observations into situation calculus device models to perform dynamical diagnosis, or (Baral, McIlraith, & Tran 2000) who do likewise in the language $\mathcal{L}$.

In this paper, we examine these issues in a dialect of the situation calculus that has been extended with knowledge-producing actions[2] (Scherl & Levesque 1993), but which does not include state constraints. Following (McIlraith 2000), we add state constraints to this language in order to

[1] Partially-Observable Markov Decision Processes (POMDPs) address this class of problems within a different formalism, but they do not address the testing issues we examine here.

[2] Henceforth referred to simply as *sensing* actions.

model the relationship between objects in the world, adopting the associated solution to the ramification problem for world-altering actions. We show that this solution extends to solve the ramification problem in the presence of sensing actions. Next, we define the notion of a test – how to design them and what knowledge can be drawn from their outcomes. In the formalization, simple tests comprise a set of initial conditions and a primitive sensing action. Complex tests are expressed as complex actions in the logic programming language Golog. We examine what it means to perform a test, and how the outcome of a test affects an agent's state of knowledge. Additionally, we examine the issue of selecting tests to confirm, refute, or discriminate a space of hypotheses.

Finally, we investigate the automation of reasoning about tests. We show that regression may be used to verify objective achievement for complex tests written in a subset of Golog. Further restrictions on the form of the complex tests allows the same regression operators to serve as the basis for a simple regression-style planner that generates tests to increase an agent's knowledge with respect to a space of hypotheses.

## Situation Calculus

The situation calculus language we use, following (Reiter 2000), is a first-order language for representing dynamically changing worlds in which all of the changes are the direct result of named *actions* performed by some agent, or the indirect result of *state constraints*. Situations are sequences of actions, evolving from an initial distinguished situation, designated by the constant $S_0$. If $a$ is an action and $s$ a situation, the result of performing $a$ in $s$ is the situation represented by the function $do(a, s)$. Functions and relations whose truth values vary from situation to situation, called *fluents*, are denoted by a predicate symbol taking a situation term as the last argument. Note that for the purposes of this paper, we assume that our theory contains no functional fluents. Finally, $Poss(a, s)$ is a distinguished fluent expressing that action $a$ is possible to perform in situation $s$. A situation calculus theory $\mathcal{D}$ comprises the following sets of axioms:

- foundational axioms of the situation calculus, $\Sigma$,
- successor state axioms, $\mathcal{D}_{SS}$,
- action precondition axioms, $\mathcal{D}_{ap}$,
- axioms describing the initial situation, $\mathcal{D}_{S_0}$,
- unique names for actions, $\mathcal{D}_{una}$,
- domain closure axioms for actions, $\mathcal{D}_{dca}$.

Successor state axioms, originally proposed by (Reiter 1991) to address the frame problem and extended by (e.g., (Lin & Reiter 1994; McIlraith 2000)) to address the ramification problem, are created by making a causal interpretation of the ramification constraints and a causal completeness assumption and compiling effect axioms of the form[3]:

$$Poss(a, s) \wedge \gamma_F^+(\vec{x}, a, s) \supset F(\vec{x}, do(a, s)) \quad (1)$$

$$Poss(a, s) \wedge \gamma_F^-(\vec{x}, a, s) \supset \neg F(\vec{x}, do(a, s)), \quad (2)$$

[3]Notational convention: all formulae are universally quantified with maximum scope unless otherwise noted.

and ramification (state) constraints of the form:

$$\upsilon_F^+(\vec{x}, s) \supset F(\vec{x}, s) \quad (3)$$

$$\upsilon_F^-(\vec{x}, s) \supset \neg F(\vec{x}, s), \quad (4)$$

into **Intermediate Successor State Axioms** of the form:

$$Poss(a, s) \supset [F_i(\vec{x}, do(a, s)) \equiv \Phi_{F_i}^*] \text{ where,} \quad (5)$$

$$\Phi_{F_i}^* \equiv \gamma_{F_i}^+(\vec{x}, a, s) \vee \upsilon_{F_i}^+(\vec{x}, do(a, s)) \vee (F(\vec{x}, s) \wedge \neg(\gamma_{F_i}^-(\vec{x}, a, s) \vee \upsilon_{F_i}^-(\vec{x}, do(a, s)))), \quad (6)$$

I.e., if an action is possible is situation $s$, then it implies that the fluent is true in $do(a, s)$ iff an action made it true -or- a state constraint made it true -or- it was already true and neither an action nor a state constraint made it false.

Such intermediate successor state axioms provide a compact representation of a solution to the ramification problem for a common class of state constraints. (McIlraith 2000) shows that for what are essentially acyclic causal ramification constraints, repeated regression rewriting (e.g., (Reiter 1991)) of $\Phi_{F_i}^*$, $\mathcal{R}^*[\Phi_{F_i}^*] = \Phi_{F_i}$, repeatedly rewrites the ramification constraints that are relativized to $do(a, s)$ in (6) above, and is guaranteed to terminate in a formula whose fluents are relativized to situation $s$ rather than $do(a, s)$. Both the intermediate and the less compact (final) successor state axioms which result from the regression provide closed-form solutions to the frame and ramification problem for the designated class of state constraints.

To illustrate sensing and testing in partially observable environments, we present a partial axiomatization of a car repair domain, derived from *The Complete Idiot's Guide to Trouble-Free Car Care* (Ramsey 1999). Our domain includes world-altering actions such as $turn\_on(x)$ and $turn\_off(x)$, where $x$ is $radio$ or $lights$. These have the effect that the radio or lights are on/off in the resulting situation. Actions $turn(key)$ and $release(key)$ have the effect that the ignition is begin turned ($turning\_ign$), or not, in the resulting situation. These actions are defined in terms of effect axioms and are combined with the following self-explanatory state constraints to produce successor state axioms. For notational convenience we abbreviate: transmission - $trans$, interlock - $intrlk$, solenoid - $solnd$, engine - $engn$, battery - $batt$; ignition system - $ign\_sys$, start system - $strt\_sys$.

$$empty(gas\_tank, s) \supset \neg startable(s) \quad (7)$$

$$ab(intrlk, s) \supset \neg startable(s) \quad (8)$$

$$ab(batt, s) \supset \neg startable(s) \quad (9)$$

$$ab(solnd, s) \supset \neg startable(s) \quad (10)$$

$$ab(starter, s) \supset \neg startable(s) \quad (11)$$

$$auto(trans) \wedge \neg ingear(trans, s) \supset ab(intrlk, s) \quad (12)$$

$$manual(trans) \wedge \neg depressed(clutch, s) \supset ab(intrlk, s) \quad (13)$$

$$turning\_ign(s) \wedge ab(batt, s) \supset \neg noise(engn, s) \quad (14)$$

$$turning\_ign(s) \wedge empty(gas\_tank, s) \supset \neg noise(engn, s) \quad (15)$$

$$turning\_ign(s) \wedge \neg ab(solnd, s) \supset noise(solnd, s) \quad (16)$$

$$ab(batt, s) \wedge on(radio, s) \supset \neg noise(radio, s) \quad (17)$$

$$ab(radio, s) \supset \neg noise(radio, s) \quad (18)$$

$$\neg ab(batt, s) \wedge on(light, s) \supset emits(light, s) \quad (19)$$

Space precludes listing all the successor state axioms. There is one (intermediate) successor state axiom for each fluent. E.g., axioms (7)–(11) compile into intermediate successor state axiom (20):

$$Poss(a, s) \supset [startable(do(a, s)) \equiv \neg empty(gas\_tank, do(a, s)) \wedge \neg ab(intrlk, do(a, s)) \wedge \neg ab(batt, do(a, s)) \wedge \neg ab(solnd, do(a, s)) \wedge \neg ab(starter, do(a, s))] \quad (20)$$

As described in (McIlraith 2000), the axioms describing the initial situation, $S_0$ contain what is known of the initial situation as well as the ramification constraints of the form of (3) and (4), relativized to $S_0$.

## Knowledge and the Ramification Problem

In (Scherl & Levesque 1993), the situation calculus language *without* state constraints was extended to incorporate both knowledge and sensing actions. World-altering actions change the state of the world, sensing actions have no effect on the state of the world but rather change the agent's state of knowledge. In our example, sensing actions include $check\_fuel$, $check\_car\_start$, $check\_radio\_noise$ etc., which have the effect of the agent knowing $empty(gas\_tank, do(a, s))$, $startable(do(a, s))$, and $noise(radio, do(a, s))$.

The notation **Knows**$(\phi, s)$ (read as $\phi$ is known in situation $s$), where $\phi$ arbitrary formula, is an abbreviation for a formula that uses $K$. For example **Knows**$(on(block_1, block_2), s)$ abbreviates:

$$\forall s'\ K(s', s) \supset on(block_1, block_2, s').$$

The notation **Kwhether**$(\phi, s)$ is an abbreviation for a formula indicating that the truth value of $\phi$ is known.

$$\textbf{Kwhether}(\phi, s) \stackrel{\text{def}}{=} \textbf{Knows}(\phi, s) \vee \textbf{Knows}(\neg\phi, s),$$

Following the notation of (Levesque 1996), each sense action $a$ has a *sensed fluent*, $SF(a, s)$ associated with it, and for each such $a$, $\mathcal{D}$ entails a sensed fluent axiom:

$$SF(a, s) \equiv \psi(s), \quad (21)$$

which says that performing the sense action $a$ tells the agent whether the formula $\psi(s)$ is true or false. Thus, $\mathcal{D} \models$ **Kwhether**$(\psi, do(s, s))$ where $a$ is an action with a sensed fluent equivalent to $\psi$.

For the sense action $check\_fuel$ the sensed fluent axiom is:

$$SF(check\_fuel, s) \equiv empty(gas\_tank, s) \quad (22)$$

which tells us whether or not the gas tank is empty. For world-altering actions, $\mathcal{D}$ entails $SF(a, s) \equiv True$.

In (Scherl & Levesque 1993), a successor state axiom for the $K$ fluent is developed. Its form is as follows:

**Successor State Axiom for** $K$

$$Poss(a, s) \supset [K(s'', do(a, s)) \equiv \exists s'.\ Poss(a, s') \wedge K(s', s) \wedge (s'' = do(a, s')) \wedge [SF(a, s') \equiv SF(a, s)] \quad (23)$$

which says that after doing action $a$ in situation $s$, the agent thinks it could be in a situation $s''$ iff $s'' = do(a, s')$ and $s'$ is a situation that was accessible from $s$, and where $s$ and $s'$ agreed on the truth value of $SF(a, s)$, e.g., the truth value of $empty(gas\_tank)$. Thus, for all situations $do(a, s)$, the $K$ relation will be completely determined by the $K$ relation at $s$ and the action $a$. This extends Reiter's solution to the frame problem (without ramifications and without knowledge) to the case of the situation calculus with sensing actions.

**Proposition 1** *In the situation calculus theory described above, the agent knows the successor state axioms and the ramification constraints.*

This follows from the fact that the successor state axioms are universally quantified over all situations, and the ramification constraints explicitly hold in $S_0$ and are entailed in all successor situations, by the successor state axioms.

**Theorem 1 (Correctness of Solution)** *The proposed solution to the frame and ramification problems for world-altering and sensing actions ensures that knowledge only changes as appropriate, as defined by Theorems 1, 2, 3 (Scherl & Levesque 1993). Furthermore, the agent knows the indirect effects of its sensing actions.*

Thus, the successor state axioms for world-altering and sensing actions, together address the frame and ramification problems.

## Testing

The purpose of a test is to attempt to determine the truth value of certain properties of the world, that may or may not be directly observable. A test is often performed with respect to a set of hypotheses, with the objective of eliminating as many hypotheses as possible from the set of hypotheses being entertained. Testing has been studied extensively for the specific problem of IC circuit testing, but there is little work on testing for rich dynamical systems such as the ones we examine here. The notion of a static test was briefly discussed in (Moore 1985, litmus example), and further developed for static systems in (McIlraith 1994; McIlraith & Reiter 1992). We build directly upon the work in (McIlraith 1994) with the objective of developing a formal theory of testing for *dynamical systems*.

Informally, a simple test comprises a set of initial conditions that may be established by the agent, together with the specification of a primitive sensing action, which determines what the agent will directly come to know as the result of the test. In our car repair domain, we can test the battery by checking the radio for noise. The initial conditions for such a test might be $on(radio, s)$. Then we can perform the sensing action $check\_radio\_noise$ to see whether the radio is emitting noise. Note that the precondition for performing the action $check\_radio\_noise$, $Poss(check\_radio\_noise, s) \equiv inside(car, s)$, is different from the initial conditions of the test. Both must hold and must be consistent with the theory and with the current hypotheses being entertained, in order to execute the test.

We distinguish between two types of tests, *truth tests* which tell us *whether* the properties being sensed are true in

the physical world, and *functional tests*, which tell us *what values* of the properties are true in the physical world. For the purposes of this paper, we restrict our attention to truth tests, and our sensing actions to so-called binary sense actions which establish the truth or falsity of a sensed formula.

**Definition 1 (Simple Test)**
*A simple test is a pair, $(I, a)$, where I, the initial conditions, is a conjunction of literals, and $a$ is a binary sense action whose sensed formula contains no free variables.*

$(on(radio, s), check\_radio\_noise)$ is an example of a simple test, following the discussion above. We now define the notion of a test for a particular hypothesis space, represented by the set $HYP$. We restrict the hypotheses, $H(s) \in HYP$ to be conjunctions of fluents whose non-situation terms are constants, and whose situation term is a situation variable $s$. In our car repair domain, an example hypothesis space might be $\{ab(batt, s), ab(solnd, s), empty(gas\_tank, s)\}$.

**Definition 2 (Test for Hypothesis Space $HYP$)**
*A test $(I, a)$ is a test for hypothesis space $HYP$ in situation $s$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for every $H(s) \in HYP$.*

That is, the state the world must be in to execute the sensing action must be satisfiable, under the assumption that any one of the hypotheses in the hypothesis space could be true. Consider that $\mathcal{D}$ entails the safety constraint $\neg explosion(s)$ and the axiom $sparks(s) \wedge gas\_leak(s) \supset explosion(s)$, and that our hypothesis space is $\{gas\_leak(s), ab(spark\_plug, s)\}$. A reasonable test for $ab(spark\_plug, s)$ is to try to create sparks at the plug. Unfortunately such a test would cause an explosion in the presence of a gas leak. The satisfiability check above precludes such a test.

**Definition 3 (Confirmation, Refutation)**
*The outcome $\alpha$ of the test $(I, a)$* **confirms** *$H(s) \in HYP$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable and $\mathcal{D} \wedge I \wedge Poss(a, s) \models$* **Knows**$(H \supset \alpha, s)$. *$\alpha$* **refutes** *$H(s)$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable and $\mathcal{D} \wedge I \wedge Poss(a, s) \models$* **Knows**$(H \supset \neg\alpha, s)$.

If the outcome of test $(on(radio, s), check\_radio\_noise)$ is $noise(radio, do(a, s))$, then our test refutes the hypothesis $ab(batt, s)$, following Axiom (17), and we can eliminate $ab(batt, s)$ from our hypothesis space, $HYP$.

Observe that a test outcome that refutes an hypothesis $H(s)$ allows us to eliminate it from $HYP$. Unfortunately, a test outcome that confirms an hypothesis is generally of no deterministic value, resulting in no reduction in the space of hypotheses. As we will see in a section to follow, there are exceptions that depend on the criteria by which the hypothesis space is defined.

In the sections to follow we use these basic definitions to define discriminating tests and relevant tests. These tests are distinguished by the effect their outcome will have on a general space of hypotheses.

## Discriminating Tests

Notice that in our example above, if we had observed $\neg noise(radio, do(a, s))$, then by the definition, this would have confirmed the hypothesis $ab(batt, s)$, but it would have been of little value in discriminating our hypothesis space. All hypotheses remain in contention. Discriminating tests are those tests $(I, a)$ that are guaranteed to discriminate an hypothesis space $HYP$, i.e., which will refute at least one hypothesis in $HYP$, regardless of the test outcome.

**Definition 4 (Discriminating Tests)**
*A test $(I, a)$ is a discriminating test for the hypothesis space $HYP$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for all $H(s) \in HYP$, and there exists $H_i(s)$, $H_j(s) \in HYP$ such that the outcome $\alpha$ of test $(I, a)$ refutes either $H_i(s)$ or $H_j(s)$, no matter what that outcome might be.*

**Proposition 2**
*After we perform a discriminating test, $(I, a)$,* **Knows**$(\neg H_j, s)$, *for some $H_j(s) \in HYP$.*

In general, we would like a discriminating test to refute half of the hypotheses in the hypothesis space, regardless of the test outcome. By definition, a discriminating test must refute at least one hypothesis in the hypothesis space.

**Definition 5 (Minimal Discriminating Tests)**
*A discriminating test $(I, a)$ for the hypothesis space $HYP$ is minimal iff for no proper subconjunct $I'$ of $I$ is $(I', a)$ a discriminating test for $HYP$.*

Minimal discriminating tests preclude unnecessary initial conditions for a test.

In some cases, we are interested in identifying a test that will establish the truth or falsity of a particular hypothesis. An individual discriminating test does precisely this.

**Definition 6 (Individual Discriminating Tests)**
*A test $(I, a)$ is an individual discriminating test for the hypotheses $H_i(s)$ and $\neg H_i(s) \in HYP$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for all $H(s) \in HYP$ and the outcome $\alpha$ of test $(I, a)$ refutes either $H_i(s)$ or $\neg H_i(s)$, no matter what that outcome might be.*

**Proposition 3**
*After we perform an individual discriminating test $(I, a)$,* **Kwhether**$(H_i, s)$ *for some $H_i \in HYP$.*

The test $(\{\}, check\_fuel)$ is such a test. The outcome will be one of $\neg empty(gas\_tank, do(a, s))$ or $empty(gas\_tank, do(a, s))$. Thus, as the result of performing $check\_fuel$ in the physical world, the agent **Kwhether**$(empty(gas\_tank, s))$.

We can similarly define the notion of a minimal individual discriminating test, and a minimal relevant test, below.

## Relevant Tests

In the majority of cases we will not be so fortunate as to have discriminating tests. Relevant tests are those tests $(I, a)$ that have the potential to discriminate an hypothesis space $HYP$, but which cannot be guaranteed to do so. Given a particular outcome $\alpha$, a relevant test may refute a subset of the hypotheses in the hypothesis space $HYP$, but may not refute any hypotheses if $\neg\alpha$ is observed. Since we can't guarantee the outcome of a test, these tests are not guaranteed to discriminate an hypothesis space. $(on(radio, s), check_radio_noise)$is an example of such a test.

**Definition 7 (Relevant Tests)**
*A test $(I, a)$ is a relevant test for the hypothesis space $HYP$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for all $H(s) in HYP$, and the outcome $\alpha$ of test $(I, a)$ either confirms a subset of the hypotheses in $HYP$ or refutes a subset.*

By definition, a relevant test confirms or refutes at least one hypothesis in $HYP$, and it follows that every discriminating test is a relevant test.

In addition to discriminating and relevant tests, there is a third class of tests. Constraining tests do not refute an hypothesis, regardless of the outcome, but they do provide further knowledge that is relevant to the hypothesis space and which the agent can exploit in combination with other tests. We discuss this notion in a longer paper.

### Testing Hypotheses

In the previous section we observed that a test outcome that refutes an hypothesis $H(s) \in HYP$ allows us to eliminate it from $HYP$, but that in general an outcome that confirms $H(s)$ has no value in reducing the hypothesis space. In this section, following (McIlraith 1994), we show that when the hypothesis space is determined using a consistency-based criterion this is indeed true, but when the hypothesis space is defined abductively, confirming test outcomes serve to eliminate those hypotheses that are not confirmed, i.e., that do not explain, the test outcome.

**Definition 8 (Consistency-Based Hypothesis Space)**
*A consistency-based hypothesis for $\mathcal{D}$ and outcome $\alpha$ of the test $(I, a)$ is any $H(s) \in HYP$ such that $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s) \wedge \alpha$ is satisfiable.*

**Proposition 4 (Eliminating C-B Hypotheses)**
*The outcome $\alpha$ of a test $(I, a)$ eliminates those consistency-based hypotheses, $H(s) \in HYP$ that are refuted by test outcome $\alpha$.*

**Definition 9 (Abductive Hypothesis Space)**
*An abductive hypothesis for $\mathcal{D}$ and outcome $\alpha$ of the test $(I, a)$ is any $H(s) \in HYP$ such that $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable, and $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s) \models \alpha$.*

**Proposition 5 (Eliminating Abductive Hypotheses)**
*The outcome $\alpha$ of a test $(I, a)$ eliminates those abductive hypotheses, $H(s) \in HYP$ that are not confirmed by test outcome $\alpha$.*

Thus, in the case of abductive hypotheses, unlike consistency-based hypotheses, both confirming and refuting test outcomes have the potential to eliminate hypotheses.

**Proposition 6 (Efficacy of Tests)**
*Any outcome $\alpha$ of a relevant test (I,a) can eliminate abductive hypotheses, whereas only a refuting outcome can eliminate consistency-based hypotheses. Discriminatory test outcomes, by definition, can eliminate either consistency-based or abductive hypotheses, regardless of the outcome.*

## Complex Tests

In the previous section, we defined the notion of a simple test $(I, a)$, and characterized the circumstances under which the outcome of such a test would discriminate an hypothesis space. Indeed, to discriminate an hypothesis space, we may need a sequence of simple tests, interleaved with world-altering actions in order to achieve the initial conditions for a test. Likewise, the selection and sequencing of sensing and world-altering actions may be conditioned on the outcome of previous sensing actions. In the section to follow, we examine the problem of generating tests using regression. As we will see, generating tests, especially tests that involve sequences of sensing and world-altering actions is hard. In many instances, we need not resort to computation. The domain axiomatizer can articulate procedures for testing aspects of a system, just as the author of *The Idiot's Guide* has done in the domain of car repair. The logic programming language, Golog (alGOl in LOGic) (Levesque *et al.* 1997) provides a compelling language for specifying such tests, as we describe briefly here.

Only a sketch of Golog is given here. See (Levesque *et al.* 1997) for a full discussion of the language and also a Prolog interpreter. Golog provides a set of extralogical constructs (such as action sequencing, if-then-else, while loops) for assembling primitive actions, defined in the situation calculus, into macros that can be viewed as complex actions. The macros are defined through the predicate $Do(\delta, s, s')$ where $\delta$ is a complex action expression. $Do(\delta, s, s')$ is intended to mean that the agent's doing action $\delta$ in situation $s$ leads to a (not necessarily unique) situation $s'$. The inductive definition of $Do$ includes the following cases:

$Do(a, s, s')$ — simple actions
$Do(\phi?, s, s')$ — tests (referred to as G-tests in this paper)
$Do([\delta_1; \delta_2], s, s')$ — sequences
$Do([\delta_1 | \delta_2], s, s')$ — nondeterministic choice of actions
$Do((\Pi x)\delta, s, s')$ — nondeterministic choice of parameters
$Do(\textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } \delta_2, s, s')$– conditionals, where we restrict $\phi$ to a G-test
$Do(\textbf{while } \phi \textbf{ do } \delta, s, s')$ — while loops

Space does not permit giving the full expansion for each of the constructs, but they can be found in (Levesque *et al.* 1997). The only change here is that the definition of the G-test construct (including the implicit G-test in the condition construct) must expand into a G-test involving knowledge[4].

The following is a partial example of a complex test written in Golog, and derived from (Ramsey 1999). This particular procedure is designed to help discriminate the space of hypotheses generated when a car won't start, namely $\{ab(intrlk, s), empty(gas\_tank, s), ab(batt, s), ab(solnd, s), ab(ign\_wires, s), ab(starter, s)\}$. In a diagnostic application such as this one, Golog procedures may also be written to combine testing with repair.

**proc** CARWONTSTART
  **if** ($\neg$ startable) **then** CHECKINTERLOCK;

[4]We are taking the simplest approach towards incorporating sensing actions into Golog. All actions are on-line. In other words, they are executed immediately without any possibility of backtracking. Other options for completely off-line execution (Lakemeyer 1999) and a mixture of off-line and on-line execution (De Giacomo & Levesque 1999a) have been discussed in the literature.

```
    if (¬ AB(INTRLK)) then CHECK_GAS_TANK;
      if (¬ EMPTY(GAS_TANK)) then CHECKBATTERY;
        if (¬ AB(BATT)) then CHECKSOLENOID;
          if (¬ AB(SOLND)) then CHECKIGNWIRES;
            if (¬ AB(IGN_WIRES)) then CHECKSTARTER;
              if (¬ AB(STARTER)) then CHECKENGINE
  end if end if end if end if end if end if end if
endProc

proc CHECKBATTERY
  TURN_ON(RADIO); CHECK_RADIO_NOISE;
    if (¬ NOISE(RADIO))
       then TURN_ON(LIGHTS); CHECK_LIGHTS
    end if
endProc
```

Observe that complex tests often involve world-altering actions which serve to establish the preconditions and initial conditions for embedded simple tests. Also observe that in achieving the preconditions or initial conditions for simple tests, these actions change the state of the world, including potentially changing the space of hypotheses. For example, if a flashlight isn't emitting light, and one hypothesis is that the batteries are dead, a good way to test them is to replace them with fresh batteries, and see whether the flashlight then works. However, replacing the flashlight batteries potentially changes the state of one of the hypotheses.

In diagnosis domains, such as the ones above, it is often desirable to combine fault detection (hypothesis testing) with repair and to take actions to eradicate faults as easily as to diagnose them (McIlraith 1997; Baral, McIlraith, & Tran 2000). However, in cases where it is desirable not to alter the truth status of the hypothesis space, care must be taken to design and verify and/or generate tests that maintain designated knowledge constraints and world constraints. E.g., we don't want to determine whether the gas tank is empty by draining it!

## Automated Reasoning About Tests

In the previous section we introduced the notion of a complex test, demonstrating that such tests could sometimes be specified in Golog. In this final technical section we briefly examine the use of automated reasoning techniques, and in particular the use of regression rewriting, for the purpose of verifying certain properties of Golog-specified complex tests, and for generating complex tests as conditional plans. Our presentation draws upon (Lespérance 1994) and (Reiter 2000). Other related approaches to conditional planning include (Rosenschein 1981; Manna & Waldinger 1987; Lobo 1998).

Consider the Golog complex test given above to help discriminate the space of hypotheses generated when a car won't start. To verify that it is an individual discriminating test, it is necessary to ensure that for at least one of the hypotheses $H$, $\mathbf{Kwhether}(H, s)$ holds, where $s$ is the situation resulting from the execution of the Golog procedure, i.e. $Do(\text{CARWONTSTART}, S_0, s)$. Thus, we would like to be able to entail $\bigvee_{H \in HYP} \mathbf{Kwhether}(H, s)$, and in particular $\mathbf{Kwhether}(empty(gas\_tank), s)$, for example. A verification that the procedure is a discriminating test would involve ensuring that for at least one $H$, $\mathbf{Knows}(\neg H, s)$ holds in the final situation, i.e., $\bigvee_{H \in HYP} \mathbf{Knows}(\neg H, s)$.

In (Scherl & Levesque 1993), a form of *regression* (based on the discussion in (Reiter 1991)) is developed for the situation calculus with sensing actions. Through the application of regression, reasoning about situations reduces to reasoning in the initial situation, $S_0$. Given a ground situation term (i.e. a term built on $S_0$ with the function *do* and ground action terms) $s_{gr}$, the problem is to determine whether the axiomatization of the domain $\mathcal{D}$ entails $G(s_{gr})$ where $G$ (the intended objective of the procedure) is an arbitrary sentence including knowledge operators. This question is reduced to the question of whether or not the axiomatization of the initial situation entails the regression of $G(s_{gr})$, i.e., $\mathcal{R}(G(s_{gr}))$. Since the result of regression is a formula in an ordinary modal logic of knowledge (i.e. a formula without action terms and where the only situation term is $S_0$) an ordinary modal theorem proving method may be used to determine whether or not the regressed formula is entailed by the axiomatization of the initial situation, $\mathcal{D}_{S_0}$. In our case $G$ will be a formula made up of subformulae of the form $\mathbf{Kwhether}(H, s)$ or $\mathbf{Knows}(\neg H, s)$, where $H$ is an hypothesis.

The regression operator $\mathcal{R}$ is defined relative to a set of successor state axioms $\mathcal{D}_{ss}$. The first four parts of the definition of the regression operator[5], $\mathcal{R}$ concern world-altering actions and are taken from (Reiter 2000).

i. When $W$ is a non-fluent atom, including equality atoms, and atoms with the predicate symbol *Poss*, or when $W$ is a fluent atom or **Knows** operator, whose situation argument is the situation constant $S_0$, $\mathcal{R}[W] = W$.

ii. When $F$ is a relational fluent (other than $K$) atom whose successor state axiom in $\mathcal{D}_{SS}$ is

$$Poss(a, s) \supset [F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F]$$

then

$$\mathcal{R}[F(t_1, \ldots, t_n, do(\delta, \sigma))] = \Phi_F|^{x_1, \ldots, x_n, a, s}_{t_1, \ldots, t_n, \delta, \sigma}$$

iii. Whenever $W$ is a formula,
$\mathcal{R}[\neg W] = \neg\mathcal{R}[W]$,
$\mathcal{R}[(\forall v)W] = (\forall v)\mathcal{R}[W]$,
$\mathcal{R}[(\exists v)W_1] = (\exists v)\mathcal{R}[W_1]$.

iv. Whenever $W_1$ and $W_2$ are formulas,
$\mathcal{R}[W_1 \wedge W_2] = \mathcal{R}[W_1] \wedge \mathcal{R}[W_1]$,
$\mathcal{R}[W_1 \vee W_2] = \mathcal{R}[W_1] \vee \mathcal{R}[W_1]$,
$\mathcal{R}[W_1 \supset W_2] = \mathcal{R}[W_1] \supset \mathcal{R}[W_1]$.

Following (Scherl & Levesque 1993), additional steps are needed to extend the regression operator to sensing actions[6]. Two definitions are needed for the specification to follow. When $\varphi$ is an arbitrary sentence and $s$ a situation term, then $\varphi[s]$ is the sentence that results from adding an extra argument to every fluent of $\varphi$ and inserting $s$ into that argument

[5] Some details are omitted here (e.g, regression of functional fluents, and the equality predicate). Also note that the formula to be regressed must be *regressable*. This concept is fully defined in (Reiter 2000).

[6] Regression of sensing actions that make known the denotation of a term (e.g. an action of reading a number on a piece of paper) is not discussed here.

position. The reverse operation $\varphi^{-1}$ is the result of removing the last argument position from all the fluents in $\varphi$.

Step **v** covers the case of regressing a world-altering action through the **Knows** operator. Step **vi** covers the cases of regressing a sensing action through the **Knows** operator. In the definitions below, $s'$ is a new situation variable.

**v.** Whenever $a$ is not a sensing action,
$$\mathcal{R}[\mathbf{Knows}(W, do(a,s))] = \mathbf{Knows}((\mathcal{R}[W[do(a,s')]])^{-1}, s).$$

**vi.** Whenever $a$ is a sensing action, where $\psi$ is a formula such that $\mathcal{D}$ entails that $\psi[s]$ is equivalent to $SF(a,s)$,
$$\mathcal{R}[\mathbf{Knows}(W, do(a,s))] = ((\psi_i(s) \supset \mathbf{Knows}(\psi_i \supset \mathcal{R}[W[do(a,s')]]^{-1}, s)) \wedge (\neg\psi_i(s) \supset \mathbf{Knows}(\neg\psi_i \supset \mathcal{R}[W[do(a,s')]]^{-1}, s))$$

An additional operator $\mathcal{C}$ needs to be defined to handle the expansion of the complex actions found in Golog, so that we can apply regression[7]. We are only considering a subset of Golog programs – those composed of simple actions, sequencing, and conditionals. We also add the empty action **noOp** or $[]$ (names for the same operation). Also note that $\pi_a(\vec{x}, s)$ stands for the preconditions of $a(\vec{x})$ as specified in the action precondition axiom, $\mathcal{D}_{ap}$, $Poss(a(\vec{x}), s) \equiv \pi_a(\vec{x}, s)$.

**viii.** $\mathcal{C}(\mathbf{noOp}, W, s) = W(s)$

**ix.** $\mathcal{C}([a(\vec{x}); \delta], W, s) = \pi_a(\vec{x}, s) \wedge \mathcal{C}(\delta, W, do(a(\vec{x}), s))$ where $a(\vec{x})$ is a ground non-sensing simple action term.

**x.** $\mathcal{C}([\mathbf{if}\ \phi(\vec{x})\ \mathbf{then}\ \delta_1\ \mathbf{else}\ \delta_2], W, s) =$
$\mathbf{Kwhether}(\phi(\vec{x}), s) \wedge$
$[\mathbf{Knows}(\phi(\vec{x}), s) \supset \mathcal{C}(\delta_1, W, s)] \wedge$
$[\mathbf{Knows}(\neg\phi(\vec{x}), s) \supset \mathcal{C}(\delta_2, W, s)]$

We are assuming that the agent is able[8] to execute the Golog test procedure. In particular, the programmer (of the test procedure) must have ensured that at the point where an $[\mathbf{if}\ \phi(\vec{x})\ \mathbf{then}\ \delta_1\ \mathbf{else}\ \delta_2]$ statement is encountered, the executing agent must $\mathbf{Kwhether}(\phi, s)$. If not, the procedure will fail.

In the following theorem (a generalization of Theorem 2 from (Lespérance 1994), recall $\mathcal{R}^*(\varphi)$ indicates the repeated regression of $\varphi$ until further applications leave the formula unchanged.

**Theorem 2** *For any Golog procedure $\delta$, consisting of simple actions, sequences, and conditionals, and $G$ an arbitrary closed regressable formula that may include knowledge operators:*

$$\mathcal{D} \models \exists s(Do(\delta, S_0, s) \wedge G(s)) \text{ iff } \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}^*(\mathcal{C}(\delta, G, S_0))$$

Theorem 2 shows it may be verified that any Golog testing routine (utilizing concatenation and conditionals) achieves its intended objective $G$ through the use of regression followed by theorem proving in the initial database. The successor state axioms ($\mathcal{D}_{ss}$) are only used in the regression procedure. This theorem can be extended to likewise verify other properties of our Golog procedures.

[7] The $\mathcal{C}$ operator introduced here is based on (but generalizes) the E operator of (Lespérance 1994).

[8] See (Lespérance *et al.* 2000) for a discussion of ability and Golog programs. Related issues are discussed in (Lespérance 1994; Lakemeyer 1999).

We can use the above regression operator as the basis for a simple conditional planning algorithm for constructing complex tests. Following (Lespérance 1994), we consider only normal form conditional plans. These are conditional plans in which the condition in a conditional (e.g. the $\phi$ in $[\mathbf{if}\ \phi(\vec{x})\ \mathbf{then}\ \delta_1\ \mathbf{else}\ \delta_2]$) must be a sensed formula. Thus we can require that prior to any conditional with the G-test $\phi$, there must be an action $a$ such that $a$ is a sensing action and $\mathcal{D} \models SF(a,s) \equiv \phi(s)$. This guarantees that the program executing the test will always $\mathbf{Kwhether}(\phi, s)$ when a conditional is encountered. For any complex test (that is executable) consisting only of concatenation and conditionals, there must be an equivalent test in this normal form.

For $i = 1, 2, 3 \ldots$, we can define the sentences $\Gamma_i$ as:

$\Gamma_0 \stackrel{\text{def}}{=} G(s)$

$\Gamma_i \stackrel{\text{def}}{=}$
$\exists a([\exists \vec{x}(a = A_1(\vec{x}) \wedge \pi_{A_1}(\vec{x}) \vee \ldots$
$\vee \exists \vec{x}(a = A_n(\vec{x}) \wedge \pi_{A_n}(\vec{x}))]$
$\wedge\ \mathcal{R}(\Gamma_{i-1}(do(a,s)))) \vee$
$\exists a([\exists \vec{x}(a = A_1^1(\vec{x}) \wedge \pi_{A_1^1}(\vec{x}) \wedge (SF(a,s) \equiv \phi_1(s)) \wedge$
$\mathcal{R}(\phi_1(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s))) \wedge$
$\mathcal{R}(\neg\phi_1(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s)))]$
$\wedge \ldots \wedge$
$\exists a([\exists \vec{x}(a = A_m^n(\vec{x}) \wedge \pi_{A_m^n}(\vec{x}) \wedge (SF(a,s) \equiv \phi_m(s)) \wedge$
$\mathcal{R}(\phi_m(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s))) \wedge$
$\mathcal{R}(\neg\phi_m(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s)))]$

Each $\Gamma_i$ is true if there is a plan of length $i$ starting in $s$ and leading to a state satisfying $G$ (Reiter 1995; Lespérance 1994). The following theorem (essentially Theorem 3 of (Lespérance 1994)) establishes the soundness and completeness of the regression-based test planning method.

**Theorem 3** *For Golog procedure $\delta$ in normal form and $G$, an arbitrary closed regressable formula that may include knowledge operators:*

$$\mathcal{D} \models \exists s(Do(\delta, S_0, s) \wedge G(s)) \text{ iff } \textit{for some } n\ \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \Gamma_0(S_0) \vee \ldots \vee \Gamma_n(S_0)$$

This regression-based finite horizon method of generating and evaluating all normal form conditional plans of greater and greater size is certainly not designed for efficiency, but the results can serve as the foundation for building more efficient regression-based complex-test planning methods, much as similar results have served as the foundation for relatively more efficient regression based planning methods (McDermott 1991; Lespérance 1994; Rosenschein 1981). In future work we will evaluate the extension of current state of the art planning techniques based on SAT and Graphplan, to address the planning problems raised in this paper (Weld 1999).

## Summary

In this paper we presented results towards a formal theory of testing for dynamical systems, specified in the language of the situation calculus. Our first contribution was to address the ramification problem for sensing actions. We then defined the notion of a test, examining how a test can be designed and how the outcome of different types of tests affect an agent's state of knowledge. The realization of many

tests in the world requires a complex sequencing of world-altering and sensing actions, whose selection and ordering is conditioned upon the outcome of previous sensing actions. We proposed specifying such complex tests in the logic programming language Golog. We then demonstrated that regression could be used both to verify the desired objective of such complex tests, and to generate tests as conditional plans under certain restrictions.

Sensing is integral to the operation of most autonomous agents. The notion of complex and simple tests introduced here extends the body of theoretical work on sensing in dynamical systems, and has practical relevance for building agents for diagnostic problem solving, plan understanding, or simply for mobile cognitive agents that need to interact in complex environments with limited sensing.

## Acknowledgments

We thank Eyal Amir for useful comments on an earlier version of this paper. Additionally, we thank Yves Lespérance for useful discussions related to this paper. This research was supported in part by NSF grant NSF 9819116, DARPA grant N66001-97-C-8554-P00004 and by NASA grant NAG2-1337.

## References

Baral, C., and Tran, S. 1998. Formalizing sensing actions: a transition function based approach. In *Proc. of AAAI 98 Fall Symposium on Cognitive Robotics*.

Baral, C.; McIlraith, S.; and Tran, S. 2000. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proc. Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*. To appear.

De Giacomo, G., and Levesque, H. J. 1999a. An incremental interpreter for high-level programs with sensing. In Levesque, H. J., and Pirri, F., eds., *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*. Berlin: Springer. 86–102.

De Giacomo, G., and Levesque, H. J. 1999b. Progression and regression using sensors. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 160–165.

Funge, J. 1998. *Making Them Behave: Cognitive Models for Computer Animation*. Ph.D. Dissertation, Department of Computer Science, University of Toronto.

Golden, K., and Weld, D. 1996. Representing sensing actions: The middle ground revisited. In *Proc. Fifth Intnl. Conf. on Principles of Knowledge Representation and Reasoning (KR'96)*.

Lakemeyer, G. 1999. On sensing and off-line interpreting in GOLOG. In Levesque, H. J., and Pirri, F., eds., *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*. Berlin: Springer. 173–189.

Lespérance, Y.; Levesque, H. J.; Lin, F.; and Scherl, R. B. 2000. Ability and knowing how in the situation calculus. *Studia Logica*. To appear.

Lespérance, Y. 1994. An approach to the synthesis of plans with perception acts and conditionals. In *Working Notes of the Canadian Workshop on Distributed Artificial Intelligence*.

Levesque, H.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. 1997. GOLOG: A logic programming language for dynamic domains. *The Journal of Logic Programming* 31:59–84.

Levesque, H. 1996. What is planning in the presence of sensing? In *Proc. Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1139–1146.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678. Special Issue on Action and Processes.

Lobo, J. 1998. COPLAS: a COnditional PLAnner with Sensing actions. In *Working Notes of the AAAI98 Fall Symposium on Cognitive Robotics*, 109–116.

Manna, Z., and Waldinger, R. 1987. How to clear a block: A theory of plans. *Journal of Automated Reasoning* 3:343–377.

McDermott, D. 1991. Regression planning. *International Journal of Intelligent Systems* 6:356–416.

McIlraith, S., and Reiter, R. 1992. On tests for hypothetical reasoning. In W. Hamscher, L. C., and de Kleer, J., eds., *Readings in model-based diagnosis*. Morgan Kaufmann. 89–96.

McIlraith, S. 1994. Generating tests using abduction. In *Proc. Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, 449–460.

McIlraith, S. 1997. *Towards a Formal Account of Diagnostic Problem Solving*. Ph.D. Dissertation, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

McIlraith, S. 2000. A closed-form solution to the ramification problem (sometimes). *Artificial Intelligence* 116(1–2):87–121.

Moore, R. 1985. A formal theory of knowledge and action. In Hobbs, J. B., and Moore, R. C., eds., *Formal Theories of the Commonsense World*. Ablex Publishing Corp. 319–358.

Ramsey, D. 1999. *The Complete Idiot's Guide to Trouble-Free Car Care*. Alpha Books.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of J. McCarthy*.

Reiter, R. 1995. CS2532S course notes. Unpublished manuscript.

Reiter, R. 2000. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. In preparation. Draft available at http://www.cs.toronto.edu/cogrobo/.

Rosenschein, S. 1981. Plan synthesis: A logical perspective. In *Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, 331–337.

Scherl, R., and Levesque, H. 1993. The frame problem and knowledge producing actions. In *Proc. Eleventh National Conference on Artificial Intelligence (AAAI-93)*, 689–695.

Shanahan, M. 1996a. Noise and the common sense informatic situation for a mobile robot. In *Proc. Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1098–1103.

Shanahan, M. 1996b. Robotics and the common sense informatic situation. In *Proc. European Conference on Artificial Intelligence (ECAI-96)*, 684–688.

Stone, M. 1998. Abductive planning with sensing. In *Proc. Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 631–636.

Weld, D. 1999. Recent advances in AI planning. *AI Magazine*.

# Diagnosing Hybrid Systems: a Bayesian Model Selection Approach

**Sheila A. McIlraith**
Knowledge Systems Laboratory
Stanford University
Stanford CA 94025
Phone: 650-723-7932, Fax: 650-725-5850
sam@ksl.stanford.edu

**Abstract**

In this paper we examine the problem of monitoring and diagnosing noisy complex dynamical systems that are modeled as hybrid systems – models of continuous behavior, interleaved by discrete transitions. In particular, we examine continuous systems with embedded supervisory controllers that experience abrupt, partial or full failure of component devices. Building on our previous work in this area (MBCG99; MBCG00), our specific focus in this paper is on the mathematical formulation of the hybrid monitoring and diagnosis task as a Bayesian model tracking and selection problem, and provision of a suitable tracking algorithm. The nonlinear dynamics of many hybrid systems present challenges to probabilistic tracking. Further, probabilistic tracking of a system for the purposes of diagnosis is problematic because the models of the system corresponding to failure modes are numerous and generally very unlikely. To focus tracking on these unlikely models and to reduce the number of potential models under consideration, we exploit logic-based techniques for qualitative model-based diagnosis to conjecture a limited initial set of consistent candidate models. In this paper we discuss alternative tracking techniques that are relevant to different classes of hybrid systems, focusing specifically on a method for tracking multiple models of nonlinear behavior simultaneously using factored sampling and conditional density propagation. To illustrate and motivate the approach described in this paper we examine the problem of monitoring and diagnosing NASA's Sprint AERCam, a small spherical robotic camera unit with 12 thrusters that enable both linear and rotational motion.

## Introduction

We have been conducting an ongoing project to investigate how to diagnose hybrid systems – complex dynamical systems whose behavior is modeled as a hybrid system. Following the description in (MBCG99; MBCG00), hybrid models comprise both discrete and continuous behavior. They are typically represented as a sequence of piecewise continuous behaviors interleaved with discrete transitions (e.g., (Bra95)). Each period of continuous behavior represents a so-called *mode* of the system. For example, in the case of NASA's Sprint AERCam, a spherical airborne robot camera unit, modes might include *translate_X-axis*, *rotate_X-axis*, *translate_Y-axis*, etc. (AG98). In the case of an Airbus fly-by-wire system, modes might include *take-off*, *landing*, *climbing*, and *cruise*. Mode transitions generally result in changes to the set of equations governing the continuous behavior of the system, as well as to the state vector that initializes that behavior in the new mode. Discrete transitions that dictate such mode switching are modeled by finite state automata, temporal logics, switching functions, or some other transition system, while continuous behavior within a mode is modeled by, e.g., ordinary differential equations (ODEs), difference equations, or differential and algebraic equations (DAEs). For the purposes of this paper, we restrict our attention to discrete-time estimation for the class of systems whose hybrid models contain no autonomous jumps. I.e., all nominal transitions between system modes are induced by a controller action; none are induced by the system state and mode (Bra95).

In (MBCG99) we presented the hybrid diagnosis problem:

> *Given a hybrid model of system behavior, a history of executed controller actions, a history of observations, including observations of aberrant behavior relative to the model, isolate the fault that is the cause for the aberrant behavior.*

Our task was to perform diagnosis online in conjunction with the continued operation of the system. Hence, we divided our diagnosis task into two stages, initial conjecturing of candidate diagnoses and subsequent refinement and tracking to select the most likely diagnoses. We cast the diagnosis problem as the problem of finding a model and associated parameter values that best fit the data. In that paper we focused on the problem of dealing with the multitude of potential models of the system by exploiting qualitative diagnosis techniques to generate a set of candidate qualitative diagnoses, and we described two parameter estimation techniques to deal with estimating the parameters associated with the model, particularly when erroneous behavior manifested itself some period of time after the initial occurrence of a fault. (See (MBCG00; MBCG99) for details.) We did not discuss the specific problem of tracking multiple candidate models, nor did we discuss how to compare them.

In this paper, we formulate the hybrid monitoring and diagnosis task as a Bayesian model tracking and selection problem (e.g., (Mac91)). In particular, we wish to estimate the state (model) of the system at successive time instants, given a history of observations. The system *diagnosis* is de-

scribed by the value of a specific subset of the state variables – namely those that designate whether components are normal or abnormal, and what their associated parameter values are. We estimate state by tracking the posterior distribution of the state, given the observations.

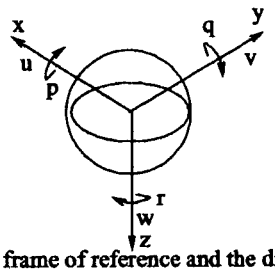
Probabilistic tracking of complex hybrid systems for diagnosis purposes presents a number of interesting challenges. Kalman filtering techniques, traditionally used for tracking linear dynamical systems with Gaussian noise, assume a Gaussian density which is unimodal, making a Kalman filter (Kal60) inadequate for simultaneously tracking alternative candidate models. Multiple Kalman filters, one for each candidate model, can sometimes be used to track multiple candidate models of linear dynamical systems with Gaussian noise (e.g., (Fra90)). More importantly, hybrid systems often have complex nonlinear, nonGaussian and potentially nondeterministic behavior. The nonlinearities come from both the mode switching (faulty or normal modes of behavior), and from the nonlinear dynamics within a mode. The latter has been addressed in some cases by using local linear (Taylor series) approximations of the nonlinear continuous dynamics, such as is done with Extended Kalman Filters (e.g., (BF88)) or Iterated Extended Kalman Filters (e.g., (Jaz70)).

In this paper, following research on bootstrap filters, particle filters and the condensation algorithm (e.g., (GSS93; IB98)), we use a factored sampling technique to sample and represent our multimodal posterior distribution of the state (models) given the observations. Such a technique enables us to track multiple models of nonlinear systems simultaneously. Unfortunately, sampling techniques for probabilistic tracking focus on the most likely models within the distribution, whereas most fault models have low probability, initially. To overcome this bias, we show how to integrate the qualitative diagnosis techniques described in (MBCG00; MBCG99) into the temporal prior of our Bayesian formulation to focus sampling on models that are indicated by our qualitative candidate diagnoses.

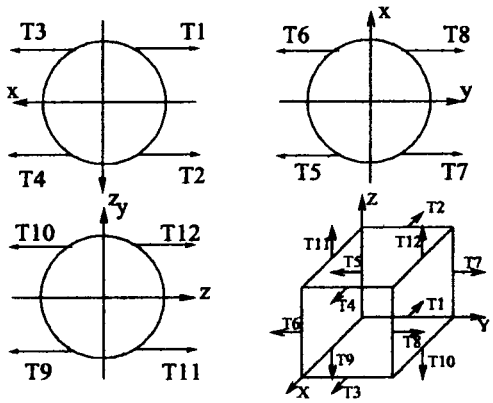
In the next section, we provide a brief description of NASA's Sprint AERCam, which we have used as a motivating example and which we will use to illustrate certain concepts in this paper. In the section that follows the description of the AERCam, we present a formal characterization of the class of hybrid systems we study and the diagnosis problem they present. Next, we describe our Bayesian formulation of the problem and the algorithm we use for computing and propagating posterior distributions. In the final section, we summarize, discuss our continuing research in this area, and reference some related work.

## The AERCam

We are using NASA's Sprint AERCam and a simulation of system dynamics and the controller written in Hybrid CC (HCC) (AG98) as a testbed for this work. To make this paper somewhat self-contained, we condense and repeat the description provided in (MBCG99). The AERCam is simpler than many of the complex systems we intend to diagnose, but it serves well in illustrating the concepts developed




The body frame of reference and the directions of velocities (u, v, w) are the components of the translation velocity. (p, q, r) are components of the angular velocity.




Three views of the AERCam, showing the thrusters, and showing all the thrusters together in the cube circumscribing the AERCam.

Figure 1: The AERCam axes and thrusters

here, and has provided an excellent testbed for our preliminary work. We describe the dynamic model of the AERCam system briefly, a more detailed description of the model and simulation appear in (AG98).

The AERCam is a small spherical robotic camera unit, with 12 thrusters that allow both linear and rotational motion (Fig. 1). For the purposes of this model, we assume the sphere is uniform, and the fuel that powers the movement is in the center of the sphere. The fuel depletes as the thrusters fire.

The dynamics of the AERCam are described in the AERCam body frame of reference. The translation velocity of this frame with respect to the shuttle inertial frame of reference is 0. However, its orientation is the same as the orientation of the AERCam, thus its orientation with respect to the shuttle reference frame changes as the AERCam rotates (i.e., it is not an inertial frame). The twelve thrusters are aligned so that there are four along each major axis in the AERCam body frame. For modeling purposes, we assume the positions of the thrusters are on the centers of the edges of a cube circumscribing the AERCam. Thus, for example, thrusters $T_1, T_2, T_3, T_4$ are parallel to the $x$-axis and are used for translation along the $x$-axis or rotation around

the $y$-axis. I.e., firing thrusters $T_1$ and $T_2$ results in translation along the positive $x$-axis, and firing thrusters $T_1$ and $T_4$ results in a negative rotation around the $y$-axis. AERCam operations are simplified by limiting them to either translation or rotation. Thrusters are either on or off, therefore, the control actions are discrete. In a normal mode of operation, only two thrusters are on at any time.

### AERCam dynamics

A simplified model of the AERCam dynamics based on Newtonian laws is derived using an inertial frame of reference fixed to the space shuttle. The AERCam position in this frame is defined as the triple $(x, y, z)$. Let $\vec{\mathbf{V}}$ be the velocity in the AERCam body frame, with its vector components given by $(u, v, w)$. The frame rotates with respect to the inertial reference frame with velocity $\omega = (p, q, r)$, the angular velocity of the AERCam. The rotating body frame implies an additional Coriolis force acting upon the AERCam. We assume uniform rotational velocity since in the normal mode of operation, the AERCam does not translate and rotate at the same time (Arn78, pg. 130). Similar equations can be derived for the rotational dynamics (AG98).

$$d(m\,\vec{\mathbf{V}})/dt = \vec{\mathbf{F}} - 2m(\vec{\mathbf{V}} \times \vec{\omega}) \quad \text{Newton's Law}$$
$$\vec{\mathbf{V}}\,dm/dt + md(\vec{\mathbf{V}})/dt = \vec{\mathbf{F}} - 2m(\vec{\omega} \times \vec{\mathbf{V}})$$

The resultant equation for each coordinate:

$$du/dt = F_x/m - 2(qw - vr) - (u/m) * dm/dt$$
$$dv/dt = F_y/m - 2(ru - pw) - (v/m) * dm/dt$$
$$dw/dt = F_z/m - 2(pv - qu) - (w/m) * dm/dt,$$

where the force $F$ on each axis, is a function of the percentage degradation of the thrusters that are exerting force in that direction as specified in Figure 1. Under normal operating conditions, the thrusters operate at 100%.

We use these models to predict the position of the AERCam at time $t+1$, given the position at time $t$. We add noise to each of the models above. In this case the noise is white Gaussian noise with a mean of zero and a standard deviation $\sigma$. As noted above, these models are implemented in HCC and are used to compute the likelihood described in the next section.

### Position Control Mode of the AERCam

In the position control mode, the AERCam is directed to go to a specified position and point the camera in a particular direction. Assume the AERCam is at position A and directed to go to position B. In the first phase, the AERCam rotates to get one set of thrusters pointed towards B. These are then fired, and the AERCam cruises towards B. Upon reaching a position close to B, it fires thrusters to converge to B, and then rotates to point the camera in the desired direction.

To facilitate the illustration of the diagnosis problem, we use a simple trapezoidal controller, which we explain in two dimensions. Suppose the task is to travel along the $x$-axis for some distance, then along the $y$-axis. Such manoeuvres are needed for navigating in the space shuttle. In order to do this, the AERCam fires its $x$ thrusters for some time. Upon reaching the desired velocity, these are switched off. When the AERCam has reached a position close to the desired $x$ position, the reverse thrusters are switched on, and the AERCam is brought to a halt — the velocity graph is a trapezium. The process is analogous for the $y$ direction.

## Problem Formulation

In this section we describe our formulation of the hybrid diagnosis problem. Once again, the hybrid systems we examine are discrete-time hybrid systems. Observations and state estimation are made at regular intervals $1, 2, \ldots, t, t+1, \ldots$. Further, we assume that our systems contain no autonomous jumps. I.e., all nominal transitions between system modes are induced by a controller action, none are induced by the system state and mode (Bra95). Autonomous jumps are common in hybrid models where a mode with complex nonlinear behavior has been simplified by creating multiple modes of less complex behavior, with state-induced autonomous jumps connecting them. Building on the concepts in (MBCG00):

**Definition 1 (Hybrid System)** A hybrid system is a 5-tuple $\langle \mathcal{M}, X, \Sigma, V, f \rangle$:

- $\mu \in \mathcal{M}$ is the discrete state or mode of the system, where $\mathcal{M}$ is a finite collection of variables. $\mu_t$ is the system mode at time $t$.
- $x \in X \subseteq R^n$ is the continuous state vector of the system. $x_t$ is the continuous state at time $t$.
- $\sigma \in \Sigma$, is the discrete input, where $\Sigma$ is a finite collection of actions. I.e., the controller actions that transition the system between modes.
- $v \in V \subseteq R^n$ is the continuous input.
- $f$ is the system dynamics function that maps the mode, the continuous state, and the input into the mode and continuous state at the next discrete time point. $(\mu_{t+1}, x_{t+1}) = f(\mu_t, x_t, \sigma_t, v_t, w_t)$, where $w_t \in R^m$ is zero-mean white noise of known pdf, and $f : \mathcal{M} \times X \times \Sigma \times R^n \times R^m \rightarrow \mathcal{M} \times X$. $f$ is often expressed as a collection of functions, e.g., functions that describe the continuous behavior within a specific mode, and a function that describes the discrete transitions between modes, based on discrete input.
- $obs \in R^p$ is the observation vector of the system. $obs_t$ is the observation vector at time $t$. $obs_t$ is related to the continuous state vector $x_t$ by the function $obs_t = h(x_t, \nu_t)$ where $\nu_t \in R^r$ is zero-mean white noise of known pdf, and $h : X \times R^r \rightarrow R^p$.

**Definition 2 (System State)** The state of a hybrid system at time $t$, $(\mu_t, x_t)$ comprises the discrete mode of the system and the continuous state at $t$.

To define the hybrid diagnosis problem, we augment Definition 1 as follows.

**Definition 3 (Diagnosable Hybrid System)** A diagnosable hybrid system, $\langle \mathcal{M}, X, \Sigma, V, f, COMPS \rangle$ is a hybrid system comprised of $m$ potentially malfunctioning components $COMPS = (c_1, \ldots, c_m)$ where

- For each $\mu \in \mathcal{M}$, $\mu$ includes a designation of whether each $c_i \in COMPS$ is operating normally, or abnormally, i.e., $[\neg]ab(c_i)$.
- For each $\mu$, continuous state vector $x$ includes a set of distinguished parameters $\theta$ associated with that mode.
- We assume that transitions to fault modes are achieved by exogenous actions. Hence, $\Sigma = \Sigma_c \cup \Sigma_e$, where
  - $\Sigma_c$ is a finite set of controller actions, and
  - $\Sigma_e$ is a finite set of exogenous actions.

We introduce the following additional notation,

- $\mathcal{O}$, designates the observation history, the sequence of time-indexed observations. $\mathcal{O}_t$ designates the observation history to time $t$.
- $\mu_F$ denotes a faulty mode, i.e., a mode for which at least one $c_i \in COMPS$ is $ab(c_i)$ in $\mu_F$. $\theta_F$ denotes the parameters associated with $\mu_F$.

In the case of the AERCam example, the potentially malfunctioning components are the 12 thrusters, and a mode $\mu$ includes the behavior mode (e.g., translate-x, translate-y, rotate-x, etc.) and $[\neg]ab(T_i), i = 1, \ldots, 12$, for each thruster. The continuous state vector includes the $x$, $y$, $z$ position of the AERCam, velocity and acceleration. The parameter values, $\theta$ associated with each $\mu$ are the percentage degradation of each of the thrusters. As we will see later on, we make a Markov assumption with respect to computing the temporal dynamics of our system. Hence all relevant state must be included explicitly in the state variables.

**Definition 4 (Model)** A model of a diagnosable hybrid systems is a time-indexed mode sequence and associated parameter values $([\mu_1, \ldots, \mu_m], [\theta_1, \ldots, \theta_m])$. The model to time $t$ is denoted $(\vec{\mu}, \vec{\theta})$ and the model at time $t$ is denoted $s_t = (\mu_t, \theta_t)$. The model is a distinguished subset of the entire system state.

In this paper we make several simplifying assumptions regarding our diagnosis task. In particular, we make a single-time fault assumption. We assume that our systems do not experience multiple sequential faults. Further, we assume that faults are abrupt, resulting in partial or full degradation of component behavior. We cast the hybrid diagnosis task as the problem of finding the most likely model for the observation history, $P(s_t \mid \mathcal{O})$, i.e, the mode and parameter values $(\mu_t, \theta_t)$ that best fit the observations over time. To do this, we appeal to a Bayesian formulation of the problem.

## Bayesian Formulation

To monitor and diagnose a hybrid system, we must compute the posterior probability distribution over models at time $t$, given the observation history. Recall, using Bayes' rule that the *posterior* is proportional to the *likelihood* times the *prior*. I.e.,

$$p(\text{model} \mid \text{observations}) \propto p(\text{observations} \mid \text{model})\, p(\text{model}).$$

Our objective is to find the posterior probability distribution over models at time $t$, $s_t$, given the observation history up to time t, $\mathcal{O}_t$. I.e, we wish to compute $p(s_t \mid \mathcal{O}_t)$.

To compute the temporal dynamics of our system, we make a Markov assumption, i.e.,

$$p(s_t \mid s_{t-1}, \ldots, s_0) = p(s_t \mid s_{t-1})$$

Further, we assume that at each time point, there is a small probability of an exogenous action, leading to a transition to a failure mode. Finally, we assume that given the current model $s_t$, the current observations $obs_t$ and previous observation history $\mathcal{O}_{t-1}$ are independent.

Hence, in order to track our hybrid system, we can compute the *posterior* distribution of the model at time $t$ given the observation history which, according to Bayes' rule and our assumptions above, is proportional to the *likelihood* of the observation at time $t$ given the model at time $t$ ($p(obs_t \mid s_t)$) and the *temporal prior*, the prediction of the current model, given the observation history up to $t-1$,($p(s_t \mid \mathcal{O}_{t-1})$. I.e.,

$$p(s_t \mid \mathcal{O}_t) = kp(obs_t \mid s_t)\, p(s_t \mid \mathcal{O}_{t-1}),$$

where $k$ ensures that the distribution integrates to one.

The likelihood of the observations given the state is easily evaluated for the AERCam following the model described in the previous section. The temporal prior, i.e., the probability of the current model given the observation history to $t-1$ depends on the posterior over models at the previous time point, $p(s_{t-1} \mid \mathcal{O}_{t-1})$ and the *temporal dynamics*, $p(s_t \mid s_{t-1})$. I.e.,

$$p(s_t \mid \mathcal{O}_{t-1}) = \int_{s_{t-1}} p(s_t \mid s_{t-1})\, p(s_{t-1} \mid \mathcal{O}_{t-1}) ds_{t-1}$$

The temporal prior expresses the probability of a particular model given the observation history up to that point. In the case of a fault diagnosis, the likelihood of a fault model will initially be very low. If we are tracking using a finite number of parallel filters, or using a factored sampling method as suggested in the next section, this may mean that we will initially not track these fault models, or alternately that we track many low probability models which is computationally expensive. In order to focus the temporal prior more quickly and accurately on the appropriate diagnostic models, we make use of qualitative diagnosis techniques.

In (MBCG00; MBCG99), we proposed to use qualitative diagnosis techniques to generate qualitative candidate diagnoses – candidate mode and parameter values that were consistent with observations $\mathcal{O}$ in some window of time.

**Definition 5 (D-tuple (MBCG00))** A D-tuple is a 4-tuple $\langle C, \mu_F, t_F, \theta_F \rangle$, where $\mu_F$ is a fault mode, $t_F$ is the time the fault mode commenced, $\theta_F$ is the parameter values associated with the fault mode behavior, and $C$ is the set of failed (abnormal) components in $\mu_F$.

**Definition 6 (Candidate Qualitative Diagnosis (MBCG00))** Given a diagnosable hybrid system with model $(\vec{\mu}, \vec{\theta})$, input history $\mathcal{I}$[1], and observation history, $\mathcal{O}$, D-tuple $\langle C, \mu_F, t_F, \theta_F \rangle$ is a candidate qualitative diagnosis iff there exists a range of parameter values $\theta_F = [\theta_l, \theta_u]$, and time range $t_F = [t_l, t_u]$ such that the occurrence of fault mode $\mu_F$ with parameter values $\theta_F$ in time range $t_F$ is consistent with $\mathcal{O}$, $\mathcal{I}$ and $(\vec{\mu}, \vec{\theta})$.

We do not repeat the diagnosis algorithms here, but refer the reader to (MBCG00; MBCG99) for details. These generated diagnoses are used to propose a set of different models to be tracked by the system. The candidate models are generated by exploiting previous work on qualitative diagnosis of continuous systems (e.g., (MB99)), adapting the authors' causal propagation algorithms to deal with the discrete state variables and mode transitions of the hybrid systems. To incorporate this so-called oracle into our Bayesian formulation, we use it to bias or focus the temporal prior. This will in turn more heavily weight the posterior for the corresponding fault models, $s_l$. In the case of particle filtering, the technique we propose in the next section to compute the posterior, this focusing of the temporal prior will help the algorithm sample from the appropriate part of the distribution. To incorporate this qualitative diagnosis "oracle" we may alter our view of the posterior we are computing as follows.

$$\begin{aligned} p(s_l \mid \mathcal{O}_l, \text{oracle}) &\propto p(obs_l \mid s_l, \text{oracle}) \\ &\quad\; p(s_l \mid \mathcal{O}_{l-1}, \text{oracle}) \\ &\propto p(obs_l \mid s_l)\, p(s_l \mid \mathcal{O}_{l-1}, \text{oracle}) \end{aligned}$$

where $p(s_l \mid \mathcal{O}_{l-1}, \text{oracle})$ is equal to $p(s_l \mid \mathcal{O}_{l-1})$ above, when the observations are consistent with the current model, and otherwise $p(s_l \mid \mathcal{O}_{l-1}, \text{oracle})$ is simply the normalized probability of the faulty models, given the observations. To ensure the speed of the oracle, and because of the lack of reliable numbers for such calculations, the probabilities generated by the oracle are normalized prior probabilities of different faults given the observations, as defined by the system builder.

Once the posterior is computed, different models can be compared by estimating the expected value of different models, normalizing and comparing. For example, we may sum the likelihoods for all samples having like $[\neg]ab(c_i)$ designations, and compare these to determine which components are likely malfunctioning.

## Computing the Posterior

In the previous section we presented the problem of tracking and diagnosing hybrid systems using a Bayesian formulation. As noted in the introduction, there are many algorithms for probabilistic tracking of dynamical systems, though most are not tailored to simultaneously tracking multiple candidate models nor to dealing with nonlinear dynamics. Our posterior distribution $p(s_l \mid \mathcal{O}_l)$ will be a multi-dimensional, multi-modal distribution, reflecting the multiple competing diagnostic models. There is no closed-form (parametric) representation for this distribution, as there is, for example, for a unimodal Gaussian. Consequently, to compute this posterior, we appeal to factored sampling techniques to provide an approximation of the distribution, and project this distribution forward through time according to its dynamics, using the Condensation algorithm (IB98), derivative of the bootstrap algorithm (GSS93) and commonly referred to as a particle filter.

[1] Previously referred to as the action history.

More specifically, the posterior distribution $p(s_l \mid \mathcal{O}_l)$, is represented as a set of $N$ weighted samples $\{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(N)}\}$, with associated weights $\{\pi^{(1)}, \ldots, \pi^{(N)}\}$. Intuitively, the larger the $N$, the better the approximation, but the more costly the computation. Hence we would like to sample the distribution as sparsely as possible, while maximizing our coverage of our distribution, and thus weighting samples more heavily in those parts of the distribution that have greater volume.

At each time step, the basic algorithm comprises three steps: select, predict, and update.

**Select:** We start with the posterior from the previous time step, $p(s_{l-1} \mid \mathcal{O}_{l-1})$, represented as the factored sample $(\mathbf{s}_{l-1}^{(i)}, \pi_{l-1}^{(i)})$, $i = 1, \ldots, N$. Sample N times with replacement with probability $\pi_{l-1}^{(i)}$, the sample $\{\mathbf{s}_{l-1}^{(i)}\}$, producing the samples $\{\mathbf{s}_l'^{(i)}\}$. Note that samples with high weights may be chosen multiple times.

**Predict:** For each new sample $\mathbf{s}_l'^{(i)}$, propagate the sample forward according to the dynamics of the system to produce new samples $\{\mathbf{s}_l^{(i)}\}$. In the case of our AERCam, these are the dynamics described in the previous section, together with zero-mean Gaussian white noise. This new set of samples approximates a fair random sample for the effective prior $p(s_l \mid \mathcal{O}_{l-1})$. What remains to compute is the weights.

**Update:** Compute the weights, $\pi_l^{(i)} = p(obs_l \mid s_l = \mathbf{s}_l^{(i)})$. From the observations $obs_l$, evaluate the likelihood of each sample, and normalize the likelihoods of the samples so they sum to 1. I.e.,

$$\pi_l^{(i)} = \frac{p(obs_l \mid \mathbf{s}_l^{(i)})}{\sum_{n=1}^{N} p(obs_l \mid \mathbf{s}_l^{(n)})}$$

The above algorithm does not reflect our qualitative diagnosis oracle. In order to suitably focus the temporal prior, we use a linear combination of the samples from the computed temporal prior, and samples from the oracle. This technique was inspired by (BF99), and could also be achieved using importance sampling.

The sample approximation to the distribution, $p(s_l \mid \mathcal{O}_l)$ can be used to compute the expected value for some moment $f$ of the density, for example a mean of some state variable, i.e.,

$$E[f(s_l) \mid \mathcal{O}_l] = \sum_{i=1}^{N} \pi_l^{(i)} f(\mathbf{s}_l^{(i)})$$

In this way, we can compare the sum of the likelihoods for each distinct model.

## Summary and Related Work

In this paper we expanded the hybrid diagnosis framework described in (MBCG99; MBCG00) to present a mathematical formulation and computational techniques for generating diagnoses of hybrid systems in terms of Bayesian tracking and model comparison. We characterized the evaluation of our models (system mode and associated parameter values)

as the computation of the posterior distribution of models given a history of observations. Exploiting a Markov assumption, we showed that this could be computed in terms of the likelihood of the observations at time $t$, given the model at time $t$, times a prior. Exploiting the work described in (MBCG99; MBCG00) for generating qualitative diagnoses of hybrid systems, we treated our qualitative monitoring and diagnosis system as an oracle. If the observations were consistent with the current model, then the qualitative monitoring and diagnosis system had no effect on the computation of the posterior. However, if the observations were inconsistent then the oracle would generate a set of candidate diagnoses that would be used to adjust the prior to focus the likelihood computation on that part of the model space that was indicated by the qualitative monitoring and diagnosis engine.

Since hybrid systems are generally nonlinear, and hence the distribution of the posterior multimodal and non-Gaussian, we represented the posterior distribution as discrete samples and exploited factored sampling techniques, used in particle filtering and in the Condensation algorithm, to propagate conditional probability densities over time.

We are still in the early stages of experimenting with these techniques, but preliminary results look promising. Condensation has proven effective for some near realtime visual tracking tasks (e.g., (IB98)), but we anticipate that more complex hybrid systems with large state spaces and partial observability will require further computation and larger amounts of memory that will compromise realtime computation, just as they do, for example, with POMDPs. Such systems will require new variants of many of the techniques we currently employ in model-based diagnosis including exploiting problem decomposition, compact representations of state spaces, abstractions of problems, and approximation of inference. In summary, Bayesian tracking and model comparison and factored sampling techniques for dynamical systems provide a sound mathematical formalism and promising tools for monitoring and diagnosing complex dynamical systems.

The problem of monitoring and diagnosing hybrid systems has received little attention to date, although there is much related work. Within the AI community, there has been a great deal of research on diagnosing static systems (e.g., (HCD92)), while much less on diagnosing discrete dynamical systems (e.g., (CT94; McI98; WN96; BLPZ99)), qualitative diagnosis of continuous systems (e.g.,. (MB99)), and tracking (e.g., (RK99)). Most recently, (LPKB00), have developed related techniques for monitoring and diagnosing *Conditional Linear Gaussian* hybrid systems using a Dynamic Bayes Nets to compactly represent the conditional probability distribution, and proposing algorithms for hypothesis reduction and smoothing. Within the FDI community, the largest proportion of research has focused on diagnosing continuous systems (e.g., (Ger98; Fra90)). These approaches have often used observer schemes and/or Kalman filters to track continuous system behavior. Diagnosis of discrete-event systems has also been studied within the FDI community (e.g, (SSLST96; Lun99)). Nevertheless, our work and the concurrent work of (LPKB00) has been the first to propose a Bayesian tracking approach to diagnosing hybrid systems. Our use of factored sampling techniques and particle filtering drawn from the statistics and computer vision communities, presents a significant contribution to a challenging problem.

## Acknowledgements

This work was funded in part by NASA grant NAG2-1337. We would like to thank David Fleet for useful discussion relating to this work.

## References

L. Alenius and V. Gupta. Modeling an AERCam: A case study in modeling with concurrent constraint languages. In *Proceedings of the CP'97 Workshop on Modeling and Computation in the Concurrent Constraint Languages*, 1998.

V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer Verlag, 1978.

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

Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*, Academic Press, Boston, 1988.

M. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995.

M.J. Black and D.J. Fleet. Probabilistic detection and tracking of motion discontinuities. In *IEEE International Conference on Computer Vision*, pp. 551-558, 1999.

M. Cordier and S. Thiébaux. Event-based diagnosis for evolutive systems. Technical Report 819, IRISA, Cedex, France, 1994.

P.M. Frank. Fault diagnosis in dynamic systems using analytic and knowledge-based redundancy: a survey and some new results. *Automatica*, vol. 26, pp. 459-474, 1990.

A. Gelb *Applied Optimal Estimation*. MIT press, Boston, 1974.

J.J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, New York, 1988.

N. Gordon, D. Salmond, A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation In *IEE Proceedings on Radar, Sonar and Navigation*, Vol 140, No 2, 107-113, 1993.

W. Hamscher, L. Console and J. de Kleer *Readings in Model-based Diagnosis*. Morgan Kaufmann, 1992.

M. Isard and A. Blake. CONDENSATION – conditional density propagation for visual tracking . In *Int. J. Computer Vision*, 29(1):5–28, 1998.

A. Jazwinski. *Stochastic Processess and Filtering Theory*. Academic Press. New York, 1970.

R.E. Kalman A new approach to linear filtering and predition problems. In *Journal of Basic Engineering*, 82:35-45, 1960.

U. Lerner, R. Parr, D. Koller, G. Biswas Bayesian Fault Detection and Diagnosis in Dynamic Systems. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'2000)*, To appear, 2000.

J. Lunze. A timed discrete-event abstraction of continuous-variable systems. *Intl. Jour. of Control*, vol. 72, no. 13, pp. 1147-1164, 1999.

MacKay, D. J. C. Bayesian interpolation. *Neural Computation* 4(3):415–447, 1991.

S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pp. 167–177, 1998.

S. McIlraith, G. Biswas, D. Clancy and V. Gupta. Hybrid Systems Diagnosis. In *Proceedings of Hybrid Systems: Computation and Control Workshop*, pp. 282–295, 2000.

S. McIlraith, G. Biswas, D. Clancy and V. Gupta. Towards Diagnosing Hybrid Systems. In *Proceedings of the Tenth International Workshop on Principles of Diagnosis (DX'99)*, pp. 193-203, 1999.

P. Mosterman and G. Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics*, 1999. vol. 29, no. 6, pp. 554-565, 1999.

P. Mosterman and G. Biswas. Building hybrid observers for complex dynamic systems using model abstractions. In *International Workshop on Hybrid Systems: Computation and Control*, Nijmegen, Netherlands, March 1999.

B. Rinner and B. Kuipers. Monitoring piecewise continuous behavior by refining trackers and models. In *Hybrid Systems and AI: Modeling, Analysis and Control of Discrete + Continuous Systems*, AAAI Technical Report SS-99-05, pp. 164–169, 1999.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. on Control Systems Technology*, vol. 4, no. 2, pp. 105-124, 1996.

B. Williams and P.P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 971–978, 1996.

# Hybrid Systems Diagnosis

Sheila McIlraith[1], Gautam Biswas[2], Dan Clancy[3], and Vineet Gupta[3]

[1] Knowledge Systems Lab, Stanford University, Stanford, CA 94305
[2] Computer Science Department, Vanderbilt University, Nashville, TN 37212
[3] Caelum Research Corporation, NASA Ames Research Center, Moffett Field, CA 94035

**Abstract.** This paper reports on an on-going project to investigate techniques to diagnose complex dynamical systems that are modeled as hybrid systems. In particular, we examine continuous systems with embedded supervisory controllers that experience abrupt, partial or full failure of component devices. We cast the diagnosis problem as a model selection problem. To reduce the space of potential models under consideration, we exploit techniques from qualitative reasoning to conjecture an initial set of qualitative candidate diagnoses, which induce a smaller set of models. We refine these diagnoses using parameter estimation and model fitting techniques. As a motivating case study, we have examined the problem of diagnosing NASA's Sprint AERCam, a small spherical robotic camera unit with 12 thrusters that enable both linear and rotational motion.

## 1 Introduction

The objective of our project has been to investigate how to diagnose hybrid systems – complex dynamical systems whose behavior is modeled as a hybrid system. Hybrid models comprise both discrete and continuous behavior. They are typically represented as a sequence of piecewise continuous behaviors interleaved with discrete transitions (e.g., [7]). Each period of continuous behavior represents a so-called *mode* of the system. For example, in the case of NASA's Sprint AERCam, modes might include *translate_X-axis*, *rotate_X-axis*, *translate_Y-axis*, etc. [1]. In the case of an Airbus fly-by-wire system, modes might include *take-off*, *landing*, *climbing*, and *cruise*. Mode transitions generally result in changes to the set of equations governing the continuous behavior of the system, as well as to the state vector that initializes that behavior in the new mode. Discrete transitions that dictate mode switching are modeled by finite state automata, temporal logics, switching functions, or some other transition system, while continuous behavior within a mode is modeled by, e.g., ordinary differential equations (ODEs) or differential and algebraic equations (DAEs).

The problem we address in this paper is how to diagnose such hybrid systems. For the purposes of this paper, we consider the class of hybrid systems that are continuous systems with an embedded supervisory controller, but whose hybrid models contain no autonomous jumps. I.e., all nominal transitions between system modes are induced by a controller action, none are induced by the system state and model [7]. The class of systems we consider can be modeled as a composition of a set of component subsystems, each of which is itself a hybrid system. We assume that the system operation is being tracked by a monitoring and observer system (e.g., [19]) that ensures that the system behavior predicted by the model does not deviate significantly from the observed

behavior in normal system operation. When observations occur outside this range, the behavior is deemed to be aberrant and diagnosis is initiated. In this paper, we consider faults whose onset is abrupt, and which result in partial or complete degradation of component behavior. The general problem we wish to address can be stated as follows: *Given a hybrid model of system behavior, a history of executed controller actions, a history of observations, including observations of aberrant behavior relative to the model, isolate the fault that is the cause for the aberrant behavior.* Diagnosis is done online in conjunction with the continued operation of the system. Hence, we divide our diagnosis task into two stages, initial conjecturing of candidate diagnosis and subsequent refinement and tracking to select the most likely diagnoses.

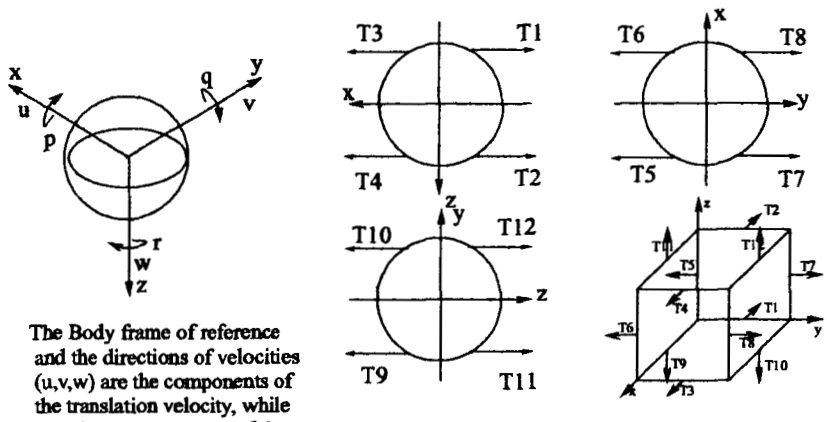
In this paper we conceive the diagnosis problem as a model selection problem. The task is to find a mathematical model and associated parameter values that best fit the system data. These models dictate the components of the system that have malfunctioned, their mode of failure, the estimated time of failure and any additional parameters that further characterize the failure. To address this diagnosis problem, we propose to exploit AI techniques for qualitative diagnosis of continuous systems to generate an initial set of qualitative candidate diagnoses and associated models, thus drastically reducing the number of potential models for our system. This is followed by parameter estimation and model fitting techniques to select the most likely mode and system parameters for candidate models of system behavior, given both past and subsequent observations of system behavior and controller actions. The main contributions of the paper are: 1) formulation of the hybrid diagnosis problem; 2) the exploitation of techniques for qualitative diagnosis of continuous systems to reduce the diagnosis search space; and 3) the use of parameter estimation and data fitting techniques for evaluation and comparison of candidate diagnoses.

In Section 2 we provide a brief description of NASA's Sprint AERCam, which we have used as a motivating example and which we will use to illustrate certain concepts in this paper. In Section 3 we present a formal characterization of the class of hybrid systems we study and the diagnosis problem they present. In Section 4 we describe our approach to hybrid diagnosis and the algorithms we use to achieve hybrid diagnosis. The generation of initial candidate qualitative diagnoses is described in Section 4.1, and the subsequent quantitative fitting and tracking of candidate diagnoses and their models is described in Section 4.2. In the final two sections, we briefly discuss related work and summarize our contributions.

## 2 Motivating Example: The AERCam

We are using NASA's Sprint AERCam and a simulation of system dynamics and the controller written in Hybrid CC (HCC) as a testbed for this work. We describe the dynamic model of the AERCam system briefly, a more detailed description of the model and simulation appear in [1].

The AERCam is a small spherical robotic camera unit, with 12 thrusters that allow both linear and rotational motion (Fig. 1). For the purposes of this model, we assume the sphere is uniform, and the fuel that powers the movement is in the center of the sphere. The fuel depletes as the thrusters fire.




The Body frame of reference and the directions of velocities (u,v,w) are the components of the translation velocity, while (p,q,r) are components of the angular velocity.

Three views of the AERCam, showing the thrusters, and showing all the thrusters together in the cube circumscribing the AERCam.

**Fig. 1. The AERCam axes and thrusters**

The dynamics of the AERCam are described in the AERCam body frame of reference. The translation velocity of this frame with respect to the shuttle inertial frame of reference is 0. However, its orientation is the same as the orientation of the AERCam, thus its orientation with respect to the shuttle reference frame changes as the AERCam rotates (i.e., it is not an inertial frame). The twelve thrusters are aligned so that there are four along each major axis in the AERCam body frame. For modeling purposes, we assume the positions of the thrusters are on the centers of the edges of a cube circumscribing the AERCam. Thus, for example, thrusters $T_1, T_2, T_3, T_4$ are parallel to the $x$-axis and are used for translation along the $x$-axis or rotation around the $y$-axis. I.e., firing thrusters $T_1$ and $T_2$ results in translation along the positive $x$-axis, and firing thrusters $T_1$ and $T_4$ results in a negative rotation around the $y$-axis. AERCam operations are simplified by limiting them to either translation or rotation. Thrusters are either on or off, therefore, the control actions are discrete. In a normal mode of operation, only two thrusters are on at any time.

### 2.1 AERCam dynamics

A simplified model of the AERCam dynamics based on Newtonian laws is derived using an inertial frame of reference fixed to the space shuttle. The AERCam position in this frame is defined as the triple $(x, y, z)$. Let $\vec{\mathbf{V}}$ be the velocity in the AERCam body frame, with its vector components given by $(u, v, w)$. The frame rotates with respect to the inertial reference frame with velocity $\omega = (p, q, r)$, the angular velocity of the AERCam. The rotating body frame implies an additional Coriolis force acting upon the AERCam. We assume uniform rotational velocity since in the normal mode of opera-

tion, the AERCam does not translate and rotate at the same time [2, pg. 130]. Similar equations can be derived for the rotational dynamics [1].

$$d(m\,\vec{\mathbf{V}})/dt = \vec{\mathbf{F}} - 2m(\vec{\mathbf{V}} \times \vec{\omega}) \quad \text{Newton's Law}$$

$$\vec{\mathbf{V}}\, dm/dt + m d(\vec{\mathbf{V}})/dt = \vec{\mathbf{F}} - 2m(\vec{\omega} \times \vec{\mathbf{V}})$$

The resultant equation for each coordinate:

$$du/dt = F_x/m - 2(qw - vr) - (u/m) * dm/dt$$
$$dv/dt = F_y/m - 2(ru - pw) - (v/m) * dm/dt$$
$$dw/dt = F_z/m - 2(pv - qu) - (w/m) * dm/dt$$

### 2.2 Position Control Mode of the AERCam

In the position control mode, the AERCam is directed to go to a specified position and point the camera in a particular direction. Assume the AERCam is at position A and directed to go to position B. In the first phase, the AERCam rotates to get one set of thrusters pointed towards B. These are then fired, and the AERCam cruises towards B. Upon reaching a position close to B, it fires thrusters to converge to B, and then rotates to point the camera in the desired direction.

To facilitate the illustration of the diagnosis problem, we use a simple trapezoidal controller, which we explain in two dimensions. Suppose the task is to travel along the $x$-axis for some distance, then along the $y$-axis. Such manoeuvres are needed for navigating in the space shuttle. In order to do this, the AERCam fires its $x$ thrusters for some time. Upon reaching the desired velocity, these are switched off. When the AERCam has reached a position close to the desired $x$ position, the reverse thrusters are switched on, and the AERCam is brought to a halt — the velocity graph is a trapezium. The process is analogous for the $y$ direction.

## 3 Problem Formulation

In this section we provide our formulation of the hybrid diagnosis problem.

**Definition 1 (Hybrid System).** A hybrid system is a 5-tuple $\langle \mathcal{M}, X, \mathcal{F}, \Sigma, \phi \rangle$, where

- $\mathcal{M}$, finite set of system modes $(\mu_1, \ldots, \mu_k)$.
- $X \subseteq R^n$, continuous state variables. $x(t)$ is the continuous behavior at time $t$.
- $\mathcal{F}$, finite set of functions $\{f_{\mu_1}, \ldots, f_{\mu_k}\}$, and associated parameter values $\theta$ such that for each mode, $\mu_i$, $f_{\mu_i}(t, \theta, x(t)) : R \times R \times X \rightarrow X$ defines the continuous behavior of the system in $\mu_i$.[1]
- $\Sigma$, finite set of actions $(\sigma_1, \ldots, \sigma_l)$, which transition the system between modes.
- $\phi$, transition function which maps an action, mode and system state vector into a new mode and initial state vector, i.e., $\phi : \Sigma \times \mathcal{M} \times X \rightarrow \mathcal{M} \times X$.

To define the hybrid diagnosis problem, we augment Definition 1 as follows.

[1] Parameter value ranges may be associated with $\theta$.

**Definition 2 (Diagnosable Hybrid System).** A diagnosable hybrid system, $\langle \mathcal{M}, X, \mathcal{F}, \Sigma, \phi, COMPS \rangle$ is a hybrid system comprised of $m$ potentially malfunctioning components $COMPS = (c_1, \ldots, c_m)$ where

- For each $\mu \in \mathcal{M}$, $\mu$ includes a designation of whether each $c_i \in COMPS$ is operating normally, or abnormally, i.e., $(\neg)ab(c_i)$.
- We assume that transitions to fault modes are achieved by exogenous actions. Hence, $\Sigma = \Sigma_c \cup \Sigma_e$, where
  - $\Sigma_c$ is a finite set of controller actions, and
  - $\Sigma_e$ is a finite set of exogenous actions.
- $\mathcal{A}$, the controller action history, the sequence of time-indexed controller actions performed.
- $X_{obs} \subseteq X$, continuous state variables that are observable. $x_{obs}(t)$ is the observations at time $t$.
- $\mathcal{O}$, the observation history, the sequence of time-indexed observations.

For notational convenience, $\mu_F$ denotes a faulty mode, i.e., a mode for which at least one $c_i \in COMPS$ is $ab(c_i)$ in $\mu_F$. $\theta_F$ denotes the parameters associated with $f_{\mu_F}$.

In the case of the AERCam example, the potentially malfunctioning components are the 12 thrusters, and a mode $\mu$ includes the behavior mode (e.g., translate-x, translate-y, rotate-x, etc.) and $(\neg)ab(T_i), i = 1, \ldots .12$, for each thruster. The continuous state vector includes the $x$, $y$, $z$ position of the AERCam, velocity and acceleration. The parameter values, $\theta$ associated with each $f_\mu$ are the percentage degradation of each of the thrusters.

**Definition 3 (Model).** A model, $Mod$ of a diagnosable hybrid systems is a time-indexed mode sequence and associated parameter values $([\mu_1, \ldots, \mu_m], [\theta_1, \ldots, \theta_m])$

Notice that each model of the system, $(\boldsymbol{\mu}, \boldsymbol{\theta})$ induces a corresponding time-indexed piecewise continuous sequence of functions $[f_{\mu_1}, \ldots, f_{\mu_m}]$ dictating system behavior.

In this paper we make several simplifying assumptions regarding our diagnosis task. In particular, we make a single-time fault assumption. We assume that our systems do not experience multiple sequential faults. Further, we assume that faults are abrupt, resulting in partial or full degradation of component behavior. We cast the hybrid diagnosis task as the problem of finding the most likely model for the observation history, $P(Mod \mid \mathcal{O})$. I.e, the sequence of modes and parameter values $(\boldsymbol{\mu}, \boldsymbol{\theta})$ that best fit the observations over time. Under normal operation, the model of the system $Mod_{normal}$ is fully dictated by the sequence of controller actions $\mathcal{A}$ and the nominal parameter values, $\theta$. Once again, we assume that the system operation is being tracked by a monitoring and observer system (e.g., [19]) that ensures that the system behavior predicted by the model does not deviate significantly from the observed behavior in normal system operation. When observations occur outside this range, the behavior is deemed to be aberrant and diagnosis is initiated. Given a diagnosable hybrid system $\langle \mathcal{M}, X, \mathcal{F}, \Sigma, \phi, COMPS \rangle$, a controller action history, $\mathcal{A}$ and a history of observations, $\mathcal{O}$ which includes observations of aberrant behavior, the **hybrid diagnosis task** is to determine what components are faulty, what fault mode caused the aberrant behavior, when it occurred, and what the values of the parameters associated with the fault mode are. In the AERCam system, a diagnosis might be that thruster $T_1$ experienced a blockage fault of 50%, at time $t_i$.

Once $Mod_{normal}$ has been rejected, we must find a new most likely model from among the potentially exponential (in $COMPS$) number of mode sequences, occurring within a large but bounded time range. We propose to exploit previous research on temporal causal graphs for qualitative diagnosis of continuous systems [18], to compute a set of candidate qualitative diagnoses that are consistent with our system, in order to identify a preliminary subset of candidate models, whose likelihood can be estimated.

**Definition 4 (D-tuple).** A D-tuple is a 4-tuple $\langle C, \mu_F, t_F, \theta_F \rangle$, where $\mu_F$ is a fault mode, $t_F$ is the time the fault mode commenced, $\theta_F$ is the parameter values associated with the fault mode behavior, and $C$ is the set of failed (abnormal) components in $\mu_F$.

**Definition 5 (Candidate Qualitative Diagnosis).** Given a diagnosable hybrid system with model $Mod = (\mu, \theta)$ an action history $\mathcal{A}$, and a history of observations, $\mathcal{O}$ which includes observations of aberrant behavior, D-tuple $\langle C, \mu_F, t_F, \theta_F \rangle$ is a candidate qualitative diagnosis iff there exists a range of parameter values $\theta_F = [\theta_l, \theta_u]$, and time range $t_F = [t_l, t_u]$ such that the occurrence of fault mode $\mu_F$ with parameter values $\theta_F$ in time range $t_F$ is consistent with $\mathcal{O}$, $\mathcal{A}$ and $Mod$.

Hence, a candidate qualitative diagnosis stipulates a fault mode, including one or more faulty components. It also stipulates a lower and upper bound, $[t_l, t_u]$, on the time the fault mode occurred. This range generally corresponds to the start times of the controller induced modes preceding and following the fault, or up to the point the fault was detected. This candidate diagnosis induces an associated *candidate model*, $Mod_C = ([\mu_1, \ldots, \mu_i, \mu_F, \mu'_{i+1}, \ldots, \mu'_m], [\theta_1, \ldots, \theta_i, \theta_F, \theta'_{i+1}, \ldots, \theta'_m])$ corresponding to $Mod$ with the fault mode $\mu_F$ and $\theta_F$ inserted at $t_F$. Every subsequent mode, $\mu_{i+1}, \ldots, \mu_m$, has $ab(c_i), c_i \in C$ enforced, and every subsequent set of parameters has the parameters associated with faulty components $C$ enforced. Computing candidate qualitative diagnoses is discussed in Section 4.1.

Since each candidate qualitative diagnosis only conjectured ranges for the time of the fault mode, $t_F$ and parameter values associated with the fault mode, $\theta_F$, the associated candidate models are underconstrained. In Section 4.2, we discuss methods for estimating unique values for $t_F$ and $\theta_F$ and for estimating a posterior probability for each of the candidate models, $Mod_C$, given $\mathcal{O}$.

**Definition 6 (Candidate Diagnosis).** Given a diagnosable hybrid system, a history of controller actions $\mathcal{A}$, and a history of observations $\mathcal{O}$, D-tuple $\langle C, \mu_F, t_F, \theta_F \rangle$ with associated model $Mod_C$ is a candidate diagnosis for the hybrid system, iff $P(Mod_C \mid \mathcal{O}) > \alpha$, for defined threshold value $\alpha \in [0.1]$.

## 4 Diagnosing Hybrid Systems

In this section we discuss one method for computing hybrid diagnoses. In Section 4.1 we discuss a technique for generating candidate qualitative diagnoses, and their associated candidate models. In Section 4.2 we discuss techniques for model fitting and for model (and hence diagnosis) comparison. In particular we discuss techniques for estimating the parameters of the candidate models, and the likelihood of the models, and for

continued monitoring and refinement of the candidate models as the system continues to operate and observations continue to be made.

We illustrate these techniques with the following simple AERCam example. Consider the scenario depicted in Fig. 2. In the first accelerate phase, the AERCam is being powered by thrusters $T1$ and $T2$. Assume that at some point in this phase, a sudden leak in the $T2$ thruster causes an abrupt change in its output. As a consequence, the AERCam starts veering to the right of the desired trajectory, as illustrated by the left-most dotted lines in Fig. 2. (The other dotted lines represent other potential candidate diagnoses consistent with the point of detection of the failure.) Soon after this occurs, the supervisory controller commands the AERCam to turn off Thrusters $T1$ and $T2$ with the objective of getting the AERCam to cruise in a straight line. In the faulty situation, the AERCam has some residual angular velocity about the z-axis, so it continues to rotate in the cruise mode. Then the controller turns on thrusters $T3$ and $T4$, to decelerate the AERCam with the objective of bringing it to a halt. Again, this objective is not entirely achieved in the the faulty situation. Next, thrusters $T5$ and $T6$ are switched on, to move the AERCam in the $y$ direction. However, since the AERCam is not in the desired orientation after the failure, the position error due to faulty thruster $T2$ accumulates causing a greater and greater deviation from the desired trajectory of the system. The position of the AERCam is being continuously sensed, filtered for noise and monitored. At some point within the $y$ translation the trajectory exceeds the error bound, i.e., $P(Mod_{normal} < \alpha)$ and is flagged by the monitoring system as aberrant relative to $Mod_{normal}$. At this point, the diagnosis task begins.

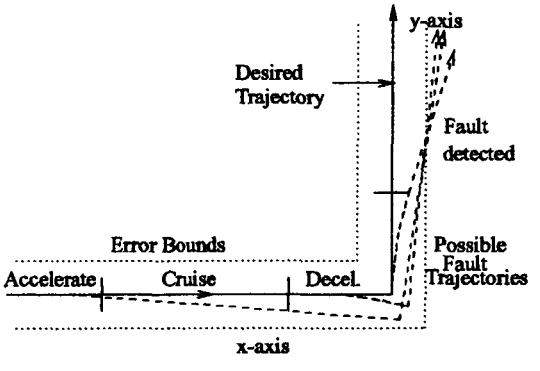



**Fig. 2.** Possible fault trajectories of AERCam ( simplified for illustration purposes).

### 4.1 Qualitative Candidate Generation

Given the current system model $Mod = (\mu, \theta)$ (commonly $Mod_{normal}$), a history of controller actions $\mathcal{A}$, and a history of observations $\mathcal{O}$ including one or more observa-

tions of aberrant behavior, we wish to generate a set of *candidate qualitative diagnoses* $\langle C, \mu_F, t_F, \theta_F \rangle$, and associated *candidate models* as described in Definition 5. To do so, we extend techniques for generating qualitative diagnoses of continuous dynamic systems to deal with hybrid systems with multiple modes. The model and propagation mechanism, as applied to continuous systems diagnosis, is described in [18].

In the case of our AERCam example, the action history $\mathcal{A}$ is [(on($T1$), on($T2$)), (off($T1$), off($T2$)), (on($T3$), on($T4$)), (off($T3$), off($T4$), on($T5$), on($T6$)), (off($T5$), off($T6$))]; the model, $Mod_{normal}$ is the time-indexed sequence [($accelerate\_x$, $\neg ab(T1-T12), \theta$), ($cruise\_x$, $\neg ab(T1-T12), \theta$),($decelerate\_x$, $\neg ab(T1-T12), \theta$), ($accelerate\_y$, $\neg ab(T1-T12), \theta$),($cruise\_y$, $\neg ab(T1-T12), \theta$)], where $\theta$ is a vector of length 12 all of whose entries are 0 (percent degradation in thrusters).

To generate candidate qualitative diagnoses we construct an abstract model of the dynamic system behavior, $Mod_{normal}$ as a temporal causal graph. A part of the temporal causal graph for the AERCam dynamics is shown in Fig. 3. The graph expresses directed cause-effect relations between component parameters and the system state variables. Links between variables are labeled as: (i) $+1$, implying direct proportionality, (ii) $-1$, implying inverse proportionality, and (iii) $\int$, implying an integrating relation. An integrating relation introduces a temporal delay in that a change on the cause side of the relation affects the derivative of the variable on the effect side. This adds temporal characteristics to the relations between variables. Some edges are labeled by variables, implying the sign of the variable in the particular situation defines the nature of the relationship. The candidate generation algorithm is invoked for every initial instance of an

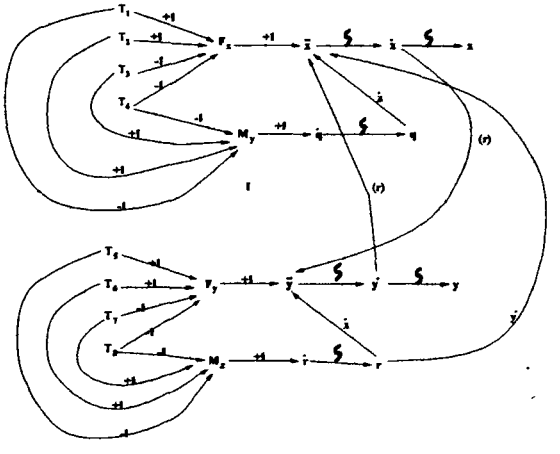


**Fig. 3.** A subset of the temporal causal graph showing the relations between Thrusters $T1 - T8$ and the x and y positions of the AERCam.

aberrant observation. The aberrant observation plus the controller action history $\mathcal{A}$ are input to a backward propagation algorithm that operates on the temporal causal graph.

The algorithm operates backwards from the last mode in the mode sequence of $Mod$:

**Step 1** For the current mode, extract the corresponding temporal causal graph model, and apply the *Identify Possible Faults* algorithm. Details of this algorithm are presented in [18], but the key aspect of this algorithm is to propagate the aberrant observation expressed as a $\pm$ value, backward depth-first through the graph. For example, given that the $y-$position of the AERCam has deviated $-$ (i.e., below normal), backward propagation implies $d(y)/dt$ is $-$, and so on, till we get $T5^-$ and $T6^-$, implying thrusters $T5$ and $T6$ are possibly faulty with decreased thrust performance. Propagation along a path can terminate if conflicting assignments are made to a node. The goal is to systematically propagate observed discrepancies backward to identify all possible candidate hypotheses that are consistent with the observations. In our example, the component parameters, $COMPS = \{T1, \ldots, T12\}$ form the space of candidate faults.

**Step 2** Repeat Step 1 for every mode in the mode sequence, to $\mu_1$. The system model needs to be substituted as the algorithm traverses the mode sequence backwards. Therefore, back propagation will be performed on a different temporal causal graph for each mode in the controller history[2].

The output of this step is a set of qualitative diagnoses $\langle C, \mu_F, t_F, \theta_F \rangle$, each with an associated candidate model, as described in Section 3. Returning to our AERCam example, three qualitative candidate diagnoses are generated. The first candidate diagnosis is that $T2$ failed in the $x$ acceleration phase. The time of the fault mode transition is $[t_1, t_2]$, and the parameters associated with the failure – the percentage degradation of the component is in the range $[0, 100]$. So the first candidate qualitative diagnosis is $\langle T2, (accelerate\_x, ab(T2), \neg ab(T1, T3 - T12), \theta_F), [t_1, t_2], [0, 100] \rangle$. The candidate model simply has $(accelerate\_x, ab(T2), \neg ab(T1), \neg ab(T3 - T12))$ inserted after the mode $(accelerate\_x, \neg ab(T1 - T12))$, and $ab(T2)$ enforced in every subsequent mode. The second candidate qualitative diagnosis is that $T4$ failed in the deceleration phase of $x$ translation, i.e., $\langle T4, (decelerate\_x, ab(T4), \neg ab(T1 - T3, T5 - T12), \theta_F), [t_3, t_4], [0, 100] \rangle$. The third candidate is that $T6$ failed during $y$ acceleration, i.e., $\langle T6, (accelerate\_y, ab(T6), \neg ab(T1 - T5, T7 - T12), \theta_F), [t_4, t_D], [0, 100] \rangle$, where $t_D$ is the time of detection of the aberrant behavior. In each case $\theta_F$ is a vector of length 12 with every entry equal to 0 (percentage degradation), except the entries corresponding to the faulty thrusters, $C$ which will have the range $[0, 100]$.

### 4.2 Model Fitting and Comparison

Given the candidate qualitative diagnoses and their associated candidate models, the next phase of the diagnosis process is quantitative refinement of the qualitative candidate diagnoses and their associated models through parameter estimation and data fitting, followed by tracking of the fit of subsequent observations to the candidate models. The goal is to at least provide a probabilistic ranking of the plausible candidates, if not a unique model (and hence diagnosis).

[2] We may cut off back-propagation along the mode sequence beyond a time limit.

As observed in the previous section, the model associated with the candidate qualitative diagnosis, $Mod_C$ is underconstrained. Both the time of the fault mode occurrence, $t_F$ and the parameters associated with the faulty behavior $\theta_F$ are represented as ranges and must be estimated. Further, the candidate qualitative diagnoses were generated from initial observations of aberrant behavior, and their consistency can be further evaluated by monitoring the qualitative transients associated with each candidate. The refinement process is performed by a set of *trackers* [21], one for each candidate diagnosis and associated model. Each tracker comprises both a *qualitative transient analysis* component and a quantitative *model estimation*, component. The two components operate in parallel as described below.

**Qualitative Transient Analysis**
The qualitative transient analysis component performs a further qualitative analysis of the consistency of candidate qualitative diagnoses based on monitoring of higher-order transients whose manifestation is seen over a longer period of time. If the transients of a candidate qualitative diagnosis do not remain consistent with subsequent observations, the candidate diagnosis will be eliminated and the *model estimation* component informed. The technique we employ is derived from techniques for qualitative monitoring of continuous systems. Details of the algorithm appear in [18].

**Model Estimation**
The purpose of the model estimation component is to perform quantitative model fitting, i.e., to provide a quantitative estimate of the parameters of the models and to assign a probability to each of the candidate models (and hence candidate diagnoses), given the noisy observed data. In particular, given a candidate model, $Mod_C$ the model estimation component uses parameter estimation techniques to estimate both the time at which the failure occurred, $t_F$, and the value for the parameters, $\theta_F$, associated with the conjectured failure mode. In this paper we discuss two alternate approaches to our time and parameter estimation problem. The first approach is based on Expectation Maximization (EM) (e.g., [8]), an iterative technique that converges to an optimal value for $t_F$ and $\theta_F$ simultaneously. The second approach we consider employs General Likelihood Ratio (GLR) techniques (e.g., [5]) to estimate the time of failure $t_F$, and then uses the observations obtained after the failure to estimate the fault parameters, $\theta_F$, by a least squares method. As described in Section 3, the outcome of both approaches is a unique value for $t_F$ and $\theta_F$ and a measure of the likelihood of $Mod_C$ given the observations. The proposed approaches to model fitting have trade-offs and we are currently assessing the efficacy of these and other alternative approaches through experimentation.

**EM-Based Approach** The Expectation Maximization (EM) algorithm (e.g., [8]) provides a technique for finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given set of data, when that data is incomplete or has missing values. The parameter estimation problem we address in this paper is a variant of the motion segmentation problem described in [24]. Here, we define the basic algorithm and the intuition behind our approach. (See [8] for more details.)

The time of failure, $t_F = [t_l, t_u]$ of our candidate qualitative diagnosisdictates the mode in which the failure is conjectured to have occurred. Let us call this mode $\mu_i$. The behavior of our hybrid system in mode $\mu_i$ is described by the continuous function

$f_{\mu_i}$, with *known* parameters $\theta_i$. At some (to be estimated) time point $t_F$ within the predicted time period of $\mu_i$, we have conjectured that the system experienced a fault which transitions it into mode $\mu_F$. The behavior of our hybrid system in mode $\mu_F$ is described by the continuous function $f_{\mu_F}$, with *unknown* parameters, $\theta_F$. We also have a set of data points $\mathcal{O}' = [x_{obs}(t_l), \ldots, x_{obs}(t_u)] \subseteq \mathcal{O}$, which either reflect the behavior of the system under $f_{\mu_i}$ or under $f_{\mu_F}$.

Given all this information, our task is to find 1) values for parameters $\theta_F$, and 2) an assignment of the data points $\mathcal{O}'$ to either $\mu_i$ or $\mu_F$ so that we maximize the fit of the data to the two functions. The assignment of data points will in turn tell us the value of $t_F$. EM provides an iterative algorithm which converges to provide a maximum-likelihood estimate for $\theta_F$ given $\mathcal{O}'$, i.e., roughly we are calculating the likelihood of $\theta$, $L(\theta) = P(\mathcal{O}' \mid \theta_F, Model)$.

The basic EM algorithm comprises two steps: an Expectation Step (E Step), and a Maximization Step (M Step) [24]:

- Select an initial (random) value for $\theta_F$.
- Iterate until convergence:
  - E Step: assign data points to either $f_{\mu_i}(\theta_i)$ or $f_{\mu_F}(\theta_F)$, which ever fits it best.
  - M Step: re-estimate $\theta_F$ using the data points assigned to $f_{\mu_F}(\theta_F)$.

The assignment of data points to $\mu_i$ and $\mu_F$ provides an estimate for $t_F$. We may exploit the fact that the assignment of data points is temporally correlated with all points before $t_f$ belonging to $\mu_i$, and all points after $t_f$ belonging to $\mu_f$. We may also exploit the fact that data points at the beginning of the interval will belong to $\mu_i$, while those at the end will belong to $\mu_F$. These task-specific qualities help our algorithm converge more quickly.

EM provides a rich algorithm for maximum-likelihood parameter estimation when we don't know the value of $t_F$. In some hybrid diagnosis applications, depending upon the sensors in our system, and the level of noise in the sensors, we may be able to develop monitoring techniques that will help isolate a reasonable value for $t_F$, minimizing the need for iteration in EM. In such cases, an alternative to the EM-based approach is to first estimate $t_F$ using the Generalized Likelihood Ratio (GLR) method [5], followed by parameter estimation of $\theta_F$.

**GLR + Least Squares Approach** Here, we divide the parameter estimation problem into two parts: (i) estimate the time of failure, $t_F$, using the Generalized Likelihood Ratio (GLR) method, and (ii) apply a standard least squares method for parameter estimation. The intuition is that solving the problem in two parts simplifies the estimation process, and very likely mitigates the numerical convergence problems that arise in dealing with complex higher-order models.

The GLR method for detecting abrupt changes in continuous signals is described in [5]. We have applied it to fault transients analysis in complex fluid thermal systems [16]. Here we provide an overview of the method for the single parameter case. Assume that the signal under scrutiny is a time-indexed sequence of random variables $y(k)$, with probability density function, $p_{\theta_i}(y)$ in desired mode $\mu_i$, and $p_{\theta_F}(y)$ in fault mode $\mu_F$. $y$ is either contained in $x_{obs}$ or computed from $x_{obs}$. We assume that a fault causes an abrupt change in $y(k)$. In the case of the AERCam, $y$ captures the difference between the observed and expected values of the, e.g., acceleration, as predicted by the model.

The central quantity in the change detection algorithm is the cumulative sum of the log-likelihood ratio for a window of observations between times $m$ and $n$,

$$S_m^n(\theta_F) = \sum_{k=m}^{n} ln \frac{p_{\theta_F}(y(k))}{p_{\theta_i}(y(k))}.$$

Again, this ratio is a function of two unknowns: $t_F$ and $\theta_F$. The common statistical solution is to use maximum likelihood estimates for these two parameters, resulting in a double maximization:

$$g_n = \max_{1 \leq m \leq n} \sup_{\theta_F} S_m^n(\theta_F).$$

If we assume that probability density functions, $p_{\theta_i}(y)$ and $p_{\theta_F}(y)$ are Gaussian, then $g_n$ reduces to:

$$g_n = \frac{1}{2\sigma_i^2} \max_{1 \leq m \leq n} \frac{1}{n-m+1} \left[ \sum_{k=m}^{n} (y(k) - \omega_i) \right]^2,$$

where $\omega_i$ and $\sigma_i^2$ are the mean and variance for $p_{\theta_i}(y)$, respectively.

When processing a sequence of samples, the point of abrupt change, $t_F$, is computed from $min\{n : g_n \geq h\}$, where $h$ is an appropriately defined threshold. Hence, the smaller the value of $h$, the more sensitive the function to change, and unfortunately to false alarms, so $h$ must be set carefully.

Once $t_F$ is estimated, data points observed after $t_F$, are used to estimate the parameter, $\theta_F$ for a hypothesized fault using regression techniques. In the case of the AERCam, the position vector of the AERCam is modeled as a set of quadratic functions in terms of the thruster force. These functions contain one unknown, $\theta_F$, the parameter that corresponds to the degree of degradation in the faulty thruster. The least squares estimate for $\theta_F$ is computed, and the the measure of fit of the candidate model to the observed data used to estimated the probability of the candidate model (and hence, diagnosis).

**Model Comparison**

From the model estimation component, each tracker computes the likelihood of its model $Mod_C$, and hence of the associated candidate diagnosis $\langle C, \mu_F, t_F, \theta_F \rangle$, as a measure of fit of the observations to the model. As new data $x_{obs}(t)$ are observed, $\theta_F$ and $t_F$, are adjusted and $P(Mod_C \mid x_{obs}(t))$ computed. If the likelihood of $Mod_C$ falls below a predefined acceptable likelihood threshold, $\alpha$, then its tracker is terminated, and the associated candidate diagnosis $\langle C, \mu_F, t_F, \theta_F \rangle$ removed from the list of candidate diagnoses. Tracking terminates when a unique diagnosis is obtained, or when the diagnoses are sufficiently discriminated to determine suitable controller actions.

## 5 Related Work

The specific problem of diagnosing hybrid systems has received little attention to date, although there is much related work. Within the AI community, there has been a great

deal of research on diagnosing static systems (e.g., [14]), while much less on diagnosing discrete dynamical systems (e.g., [17,25]), and qualitative representations of continuous systems (e.g.,. [18]). Within the FDI community, the largest proportion of research has focused on diagnosing continuous systems (e.g., [13, 11]). The most common model-based approaches use observer schemes(e.g., [12, 20]), where the goal is to design residual generators based on observed discrepancies, such that individual residuals are sensitive to a particular subset of faults. There is also complementary work by Basseville [4], using model-based statistical processing techniques for early fault detection and residual identification. [18] perform residual generation and analysis task in a qualitative framework to address some of the computational issues that arise in handling the complex dynamics that occur in fault transients, with some preliminary work on building multiple observers for hybrid systems [19]. Diagnosis of discrete-event systems has also been studied within the FDI community (e.g, [22, 15]). Fabre et al. [10] have employed stochastic Petri nets based on a Hidden Markov Model probabilistic scheme for alarm analysis. Unfortunately, it is not clear how to systematically derive such representations from the physical system models that we work with.

## 6 Summary

In this paper we addressed the problem of diagnosing hybrid systems. The main contributions of the paper are 1) formulation of the hybrid diagnosis problem as model selection; 2) the exploitation of techniques for qualitative diagnosis of continuous systems to reduce the diagnosis search space; and 3) the use of parameter estimation and data fitting techniques for evaluation and comparison of candidate diagnoses. This work continues with experimental analysis of the proposed techniques, and a more formal characterization of our approach in terms of Bayesian model selection.

## Acknowledgements

This work was funded in part by NASA grant NAG 21337. The first author would like to thank David Fleet for useful discussion relating to this work.

## References

1. L. Alenius and V. Gupta. Modeling an AERCam: A case study in modeling with concurrent constraint languages. In *Proceedings of the CP'97 Workshop on Modeling and Computation in the Concurrent Constraint Languages*, 1998.
2. V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer Verlag, 1978.
3. P. Baroni, G. Lamperti, P. Pogliano and M. Zanella Diagnosis of large active systems *Artificial Intelligence*, 110(1):135–183, 1999.
4. M. Basseville. On-board component fault detection and isolation using a statistical local approach. *Automatica*, vol. 34, no. 11, 1998.
5. M. Basseville and I.V. Nikiforov. *Detection of Abrupt Changes: Theory and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

6. J. A. Blimes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report TR-97-021, International Computer Science Institute (ICSI) and Computer Science Division, Dept. of Electrical Engineering and Computer Science, U.C. Berkeley, 1998.
7. M. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995.
8. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data. *Journal of the Royal Statistical Society Ser. B*, 39:1–38, 1977.
9. B. Etkin and L. D. Reid. *Dynamics of Flight:Stability and Control.* John Wiley and Sons, 1995.
10. E. Fabre, A. Aghasaryan, A. Benveniste, R. Boubour and C. Jard. Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets. *Journal of Discrete Event Dynamic Systems,* vol. 8, no. 2, pp. 203-231, 1998.
11. P.M. Frank. Fault diagnosis in dynamic systems using analytic and knowledge-based redundancy: a survey and some new results. *Automatica,* vol. 26, pp. 459-474, 1990.
12. E.A. Garcia and P.M. Frank. Deterministic nonlinear observer-based approaches to fault diagnosis: a survey. *Control Engineering Practice*, 5(5):663–670, 1999.
13. J.J. Gertler. *Fault Detection and Diagnosis in Engineering Systems.* Marcel Dekker, New York, 1988.
14. W. Hamscher, L. Console and J. de Kleer *Readings in Model-based Diagnosis.* Morgan Kaufmann, 1992.
15. J. Lunze. A timed discrete-event abstraction of continuous-variable systems. *Intl. Jour. of Control,* vol. 72, no. 13, pp. 1147-1164, 1999.
16. E.J. Manders, P.J. Mosterman, and G. Biswas. Signal to symbol transformation techniques for robust diagnosis in transcend. In *10th Int. Workshop on Principles of Diagnosis*, pp. 155–165, 1999.
17. S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pp. 167–177, 1998.
18. P. Mosterman and G. Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics*, 1999. vol. 29, no. 6, pp. 554-565, 1999.
19. P. Mosterman and G. Biswas. Building hybrid observers for complex dynamic systems using model abstractions. In *International Workshop on Hybrid Systems: Computation and Control*, Nijmegen, Netherlands, March 1999.
20. R.J. Patton and J. Chen. Observer-based fault detection and isolation: robustness and applications. *Control Engineering Practice,* 5(5):671–682, 1997.
21. B. Rinner and B. Kuipers. Monitoring piecewise continuous behavior by refining trackers and models. In *Hybrid Systems and AI: Modeling, Analysis and Control of Discrete + Continuous Systems*, AAAI Technical Report SS-99-05, pp. 164–169, 1999.
22. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. on Control Systems Technology,* vol. 4, no. 2, pp. 105-124, 1996.
23. W. Sweet. The glass cockpit. *IEEE Spectrum*, pages 30–38, September 1995.
24. Y. Weiss. Motion segmentation using EM – a short tutorial. http://www-bcs.mit.edu/people/yweiss/tutorials.html, 1997.
25. B. Williams and P.P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 971–978, 1996.