Model-Based Diagnostics and Recovery: an Integrated Approach

Gregory Provan
Rockwell Science Center
1049 Camino dos Rios, Thousand Oaks, CA 91360

Abstract

We describe an approach that integrates diagnostics and recovery using a single model and suite of algorithms. We extend model-based diagnosis representations and algorithms to enable control and recovery given anomalous sensor data. We show how we can use focusing mechanisms to efficiently search the space of diagnoses and of recovery actions. We use our framework to prove results about the quality of the recovery actions that can be computed, and the complexity of computing these actions.

1 Introduction

We introduce an approach that enables both diagnosis and recovery to be performed using a single system model for feedback-control systems. By diagnosis we mean isolating the cause of anomalous sensor readings, and by recovery we mean identifying appropriate control actions that can modify the anomalous operating conditions to achieve system objectives, if possible.

Typically, diagnosis and recovery tasks are performed independently, with distinct models designed for each task. For example, for diagnosis one of several model-based diagnosis approaches can be used (GDE [de Kleer and Williams,1989] or causal networks [Darwiche,1995]), or for recovery one of several approaches can be used (e.g., [Johnson,1996],[Rauch,1995]).

We extend the causal network model-based diagnostics approach [Darwiche, 1995,1998] to incorporate recovery. This extension provides this approach with a powerful new set of capabilities, in addition to providing a way to accept inputs from existing external components, such as control and planning components. Our extension can be used for any model-based diagnosis system that uses assumables for mode specification and a focusing algorithm for computing least-cost diagnoses, e.g., the class of approaches described in [Dressler and Struss,1996].

Only simple modifications to a diagnostic model are necessary to enable computation of recovery actions. We need to specify a set of control actions, together with an algebra that defines relationships among these controls; e.g., to specify the least-cost settings we can use (a) finite state automata to characterize the allowable weighted transitions from one control setting to another or (b) a partial ordering over control settings. We assign a partial ordering over control settings to facilitate a best-first search algorithm in searching for least-cost settings. In [Darwiche and Provan,1996], diagnosis models were developed for control systems for domains such as factory automation. We augment that approach by assigning weights to control actions and defining an algorithm to compute the least-cost control actions given a system fault.

Using this approach we can define two "views e-pending on the task.

1. Diagnosis/Control: this is the view when the system is operating nominally. Given anomalous sensor readings, the system will compute a least-cost mode assignment to "explain" those sensor readings.
2. Reconfiguration: when a diagnosis D is computed, the system uses a dual reconfiguration model to compute a least-cost control action given the input of the fixed mode assignment D. Hence, we do not need to pre-program all control conditions for anomalous states; they can be computed using this recovery algorithm.

Note that we do not address the planning that may be necessary should the goal not be satisfiable under a given fault scenario; we focus entirely on the diagnosis and recovery aspects.

Our contributions are as follows:

1. We describe how to extend a "traditional" diagnostic model with information necessary to perform recovery actions given a fault, such that we have a single integrated model for both diagnosis and recovery. This enables recovery capabilities similar to [Williams and Nayak,1996], but using standard model-based diagnostic approaches.
2. We describe how to use a traditional model-based diagnostic inference algorithm with focusing to efficiently compute least-cost recovery actions; as such, this framework allows us to provide guarantees for recovery similar to those possible for diagnoses.
3. We have used an open architecture such that several additional representations and algorithms can communicate with the diagnostic/recovery system using CORBA/COM protocols.

The remainder of the paper is organized as follows. In Section 2 we introduce a simple rocket engine example that we use throughout the document. In Section 3 we summarize our diagnostic approach, and show how we extend it for computing recovery actions. In Section 4 we apply our diagnostics and recovery approach to the rocket engine. We conclude by discussing related work and summarizing our contributions.

2 Simple Engine Example

This section introduces a simple engine example that will be used throughout the document. This rocket engine subsystem is intended solely for illustrative purposes, and abstracts elements common to many rocket engines, such as the Space Shuttle main engine. We use a simplified model to illustrate the concepts proposed in this document.

2.1 Engine Schematics

This engine subsystem is configured as shown in Figure 1. This figure shows that there are two inputs, liquid oxygen (LOX) and hydrogen (H_2), that are fed into the nozzle after their pressure has been increased via pumps. The LOX and H_2 are burned, and the subsystem produces a desired thrust to drive the rocket.

For each of the LOX and H_2 inputs, there is a redundant channel; we call the primary channel the A-channel, and the redundant channel the B-channel. If there are problems delivering sufficient LOX and H_2 at a high enough pressure via the A-channel, the B-channel can be used, in conjunction with or instead of the A-channel.

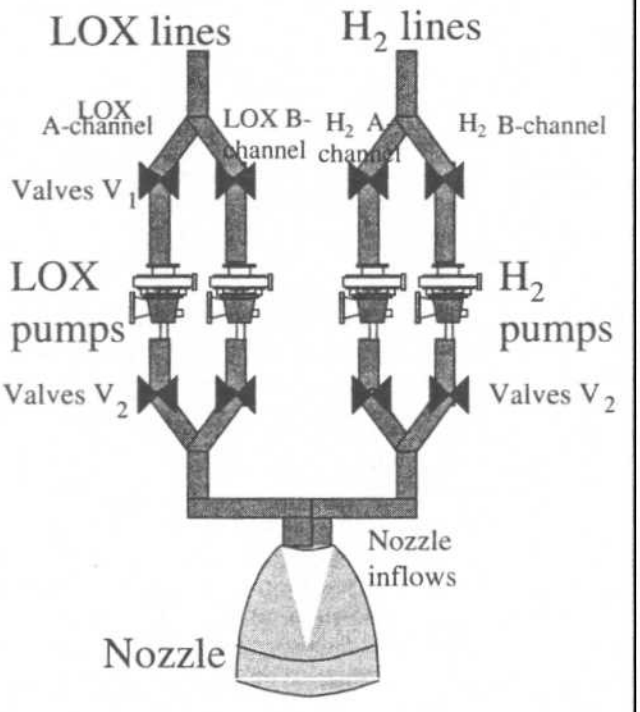

Figure 1: Schematic for rocket engine example

There are a number of valves on this subsystem to regulate the gas flow. We assume that each valve has a sensor to measure the inlet pressure. Hence the LOX and H_2 inlet-pressures into the subsystem and the valve settings are the control parameters for the subsystem.

The components that can fail are the valves, the pumps, and the nozzle. We assume that all other components cannot fail.

2.2 Engine Functionality

This section outlines the functionality of this engine. We use a qualitative representation of functionality based on multi-valued propositional relations.

The entities defined in this model are:

- Physical components: valve, sensor, nozzle;
- Pressures: LOX-pressure, H_2-pressure;
- Thrust.

We assume that all valves, as well as all sensors, function identically. Table 1 provides the possible values for some of these entities:

Entity	Possible Values
Valve	{closed, 25%-open, 50%-open, open}
Sensor	{high, nominal, low, very-low}
Pressure	{high, nominal, low, very-low}
Thrust	{100%, 75%, 50%, 0%}

Table 1: values for entities in functional model

Table 2 summarizes the mode assignments for entities we are interested in diagnosing:

Entity	Mode Assignments
Valve	{nominal, stuck-open, stuck-closed}
Pump	{nominal, low-output, failed}
Nozzle	{nominal, minor-leak, major-leak}

Table 2: Mode assignments for entities to be diagnosed

Any pressure P_V downstream of a valve has functional relationship as follows:
$P_V = \xi$(inlet-pressure, Valve-setting, Valve-mode).

Any pressure P_P downstream of a pump has functional relationship as follows:
$P_P = \xi$(inlet-pressure, Pump-setting, Pump-mode).

The thrust T is a function of the LOX and H_2 inlet-pressures and nozzle-mode:
$T = \xi$(LOX-inlet-pressure,H_2-inlet-pressure, nozzle-mode).

Figure 2 shows a causal network model of this system, identifying these causal relationships. Table 3 presents a table describing some of the relations for pump pressure, P_P. This table shows how specific values of P_P are obtained by the values of the variables on which it depends.

P_P	P_{in}	Pump-setting	Pump-mode
High High Nominal low	High Nominal Low Very-low	nominal	nominal
High High High Nominal	High Nominal Low Very-low	High-speed	nominal
High Nominal Low Very-low	High Nominal Low Very-low	nominal	Low-output

Table 3: Subset of Relations for Pump Pressure

3 Diagnostics and Recovery Framework

This section reviews the causal network model-based diagnosis framework, and then describes how we extend this framework to incorporate recovery. In summary, we add to the diagnosis framework a control specification, in addition to a control algebra that allows us to focus our search on the least-cost control actions necessary to recover from anomalous states.

3.1 Model-Based Diagnostics

The well-known model-based diagnosis framework of Reiter [1987] using the triple (SD,COMP,OBS), where SD is a system description, COMP is a set of components, and OBS is a system observation. We use a variation of this formalism, as described below.

We define a system description SD for a diagnostic model using the tuple $(\mathbf{V},\mathbf{A},\Psi)$:

Definition 1: A *system description* is defined using a tuple $(\mathbf{V},\mathbf{A},\Psi)$, where $\mathbf{V}\cup\mathbf{A}$ is the set of system variables, and Ψ is a quantification of the variables, i.e., a set of multi-valued propositional sentences constructed from $\mathbf{V}\cup\mathbf{A}$.

We specify the details of this definition as follows.

Definition 2 [System Variables]: We partition the system variables into two sets: $\mathbf{V}$ is a set of atomic propositions, $\mathbf{A}$ is a set of distinguished atomic propositions called assumables. The atomic propositions $\mathbf{V}$ describe the entities in the model, which include system components. The assumables are the variables describing the operating characteristics of the components that we want to diagnose.

We further partition the atomic propositions $\mathbf{V}$ into a set O of observable variables, and a set U of unobservable variables: $\mathbf{V} = \mathrm{U}\cup\mathrm{O}$. The observable variables are the variables that represent sensors and actuators, i.e. components whose values we can measure (sensor) or components whose values we can set (actuator). A system observation OBS is an instantiation of the values of the observables.

If each element of $\mathbf{A}$ consists only of binary-valued variables, i.e., OK and broken (or $\neg$OK), then we say that $\mathbf{A}$ is mode-less. If the elements of $\mathbf{A}$ can have finite domains, e.g., {OK,failed-high,failed-low}, then we say that $\mathbf{A}$ has modes, and we are performing diagnosis with modes. In this case each system failure-mode can exhibit multiple normal and multiple fault modes.

Definition 3 [Quantification]: Ψ is a set of propositional sentences constructed from atoms in $\mathbf{V}$ and $\mathbf{A}$. More specifically, we associate with each variable X in $\mathbf{V}$ a subset Ψ_X of sentences in Ψ, such that X is the consequent of each sentence in Ψ_X, and the antecedents of each sentence in Ψ_X are defined using the parents of X in an underlying directed acyclic graph G. This graph defines the causal relations among the variables.

The equations define a set of constraints over the possible values of each variable. We can specify a model in a number of ways, such as using logic [de Kleer and Williams,1989], constraints [Struss,1992], or causal networks [Darwiche,1995]. In this paper we define the system model using causal networks, which specify a graph G and quantification consisting of a set of multi-valued propositional clauses.

Model-based inference is as follows: given SD$\cup$OBS, find the failure-mode specification A* such SD$\cup$OBS$\cup$A* does not entail $\perp$. Here A* is an assignment of mode values to all system failure-modes (or more generally behavioral modes). Computing the value of A* is an NP-hard task, since it searches over all subsets of mode-instantiations. Since straightforward search of the space of diagnoses can be of exponential size, we apply a focusing algorithm to conduct a best-first search over the space of diagnoses [Darwiche, 1995]. This focusing algorithm assigns a set Σ of costs to the fault modes, and then uses an algebra to perform the best-first search over the diagnosis space. Σ is derived from the failure history, e.g., component reliability data. Following [Darwiche,1995], we can characterize the algebra using the triple $(\Sigma,\oplus,\geq_\oplus)$, where $\oplus$ is a cost addition operation and $\geq_\oplus$ is a cost total ordering satisfying:

- $\oplus$ is commutative, associative and has a zero element;
- $j\geq_\oplus i$ iff $i\oplus k=j$ for some k.

An example of this is $\langle \mathbf{Z},+,\geq\rangle$.

The full specification for the diagnosis task requires the tuple $\langle \Psi, \mathrm{OBS}, \mathrm{Dx\text{-}Alg}, (\Sigma,\oplus,\geq_\oplus)\rangle$, where

- Ψ is the system model,
- OBS is the input observation,
- Dx-Alg is the diagnosis inference algorithm, and
- $(\Sigma,\oplus,\geq_\oplus)$ characterizes the focusing algorithm.

3.2 Extensions for Recovery

We extend the causal network approach to diagnostic inference by creating a dual causal network for recovery. This involves reconfiguring some nodes in the network, but preserving the structure of the network. We define a recovery network by partitioning the observables O into

a set χ of control variables and a set S of sensors. In the diagnostic model, we have observables $\chi \cup S$; for a recovery model, we now have observables $A \cup S$.

We introduce the notion of a dual model to represent the recovery model. Table 4 shows the relationship between the primal (diagnosis) model and the dual (recovery) model. The key change is that we have swapped some observables (the control variables χ) and assumables **A**. We now assign a different set Σ' of costs to the control variables, and then use a different algebra to perform the best-first search over the control space.

	Variables Obs.	Unobs.	Assumables	Graph Structure
Diagnosis	$\chi \cup S$	U	**A**	G
Recovery	$A \cup S$	U	χ	G

Table 4: Relationship between primal and dual models

We can characterize the control algebra using the triple $(\Sigma', \circledR, \geq_{\circledR})$, where $\circledR$ is a cost addition operation and $\geq_{\circledR}$ is a cost total ordering satisfying the properties defined in the previous section. There are many ways that we can define this control algebra, and we outline two now.

One option is to compute the costs of control settings using Σ', and then use the same focusing algebra as done for diagnostic focusing. Our goal is to compute minimal-cost control settings that satisfy the goal Θ. In a cost assignment, we assign cost as an inverse function of component life: a cost of 0 means long life, whereas larger costs mean relatively shorter life. For example, in Table 5, the weights are assigned as follows: an open or closed valve has long life (weight 0), whereas the lifetime of the valve decreases the more it is partially opened, due to the force of fluid exerted on the valve mechanism.

Valve Setting	closed	25% open	50% open	open
Cost	0	1	2	0

Table 5: Cost assignment to valve settings

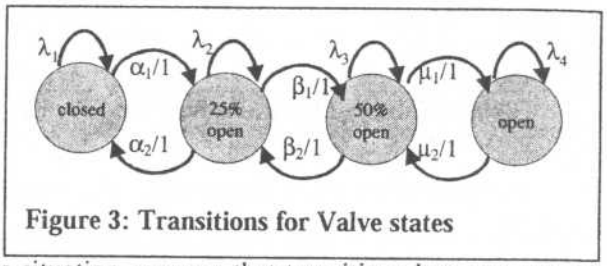

Figure 3: Transitions for Valve states

This situation assumes that transitions between any settings are possible. If, however, we want to constrain possible transitions between valve settings, then we can use a weighted automata model, as shown in Figure 3, or causal network fragments, using either a full temporal logic, or a simplification thereof. In Figure 3, all transitions λ_i, i=1,..,4, have weight 0, and the other

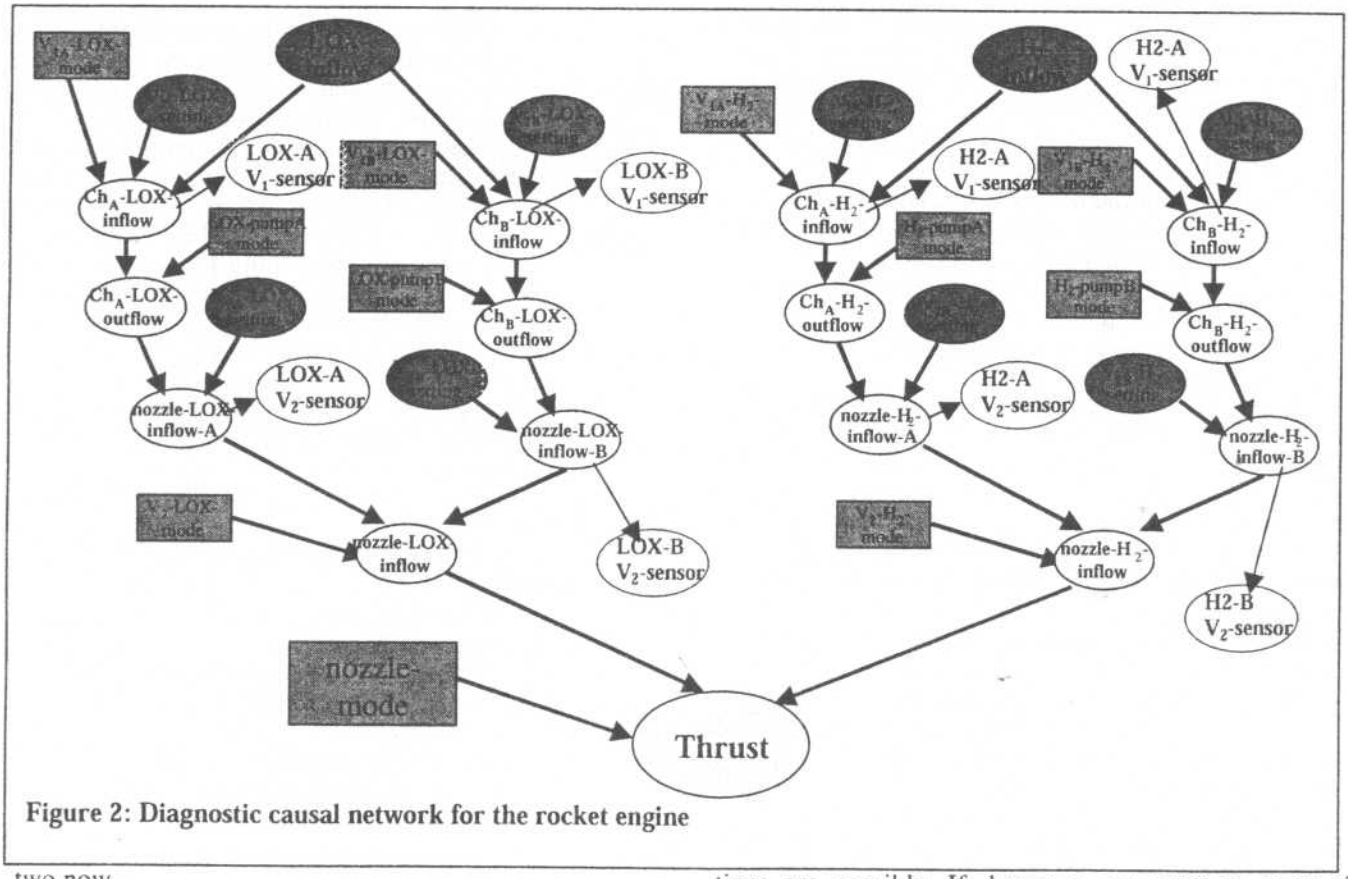

Figure 2: Diagnostic causal network for the rocket engine

transitions have weights as shown in the figure, e.g., transition α_2, which has weight 1, is denoted by $\alpha_2/1$. In the full temporal logic [Darwiche and Provan,1996], we have temporally-indexed variables and can model variables over discrete time steps. For computing reconfiguration, we can use a simplification of this approach, assuming that we only need to know the previous and current value of a variable. Specifically, if we denote the valve by variable V, and define a variable V^τ for the previous value, we can model the transitions of the Valve states using equations such as:

$[V^\tau = \text{closed}] \wedge \alpha \quad \Rightarrow [V=25\%\text{-open}]$
$[V^\tau = \text{closed}] \wedge \lambda_1 \quad \Rightarrow [V=\text{closed}]$
$[V^\tau = 25\%\text{-open}] \wedge \beta \quad \Rightarrow [V=50\%\text{-open}]$
$[V^\tau = 25\%\text{-open}] \wedge \lambda_2 \quad \Rightarrow [V=25\%\text{-open}].$

In this approach the transitions (α, λ_1, etc.) correspond to variables taking on particular values.

Using this approach, the allowable transitions add an extra constraint to the focusing algorithm for recovery. In other words, when searching for a least-cost control action, the search is constrained to the allowable set of transitions from any given place in a transition diagram. In Figure 3, given that a valve is closed, we can only keep the valve closed (transition λ_1) or open it to the 25%-open state (transition α). Three transitions (α,β,γ) are required to move the valve from the closed to the open state.

We also require the notion of a control goal, and a means of encoding that in the model. For our example, the goal is the required thrust for the rocket, and the thrust is a function of the LOX and H_2 inlet-pressures, as well as the mode of the nozzle:

Thrust = ξ(LOX-inlet-pressure, H_2-inlet-pressure, nozzle-mode).

If Thrust can take on the values {100%, 75%, 50%, 0%}, then we can see that a control goal of at least 75% thrust means that unacceptable values are 50% and 0%. This information is critical input to the recovery algorithm.

The full specification for the recovery task requires the tuple $\langle \Psi,\ \text{GOAL} \cup A^*, \text{Dx-Alg}, (\Sigma', \circledR, \geq_{\circledR}) \rangle$, where

- Ψ is the system model,
- GOAL$\cup A^*$ is the control goal together with the current setting of (faulty) assumables,
- Dx-Alg is the diagnosis inference algorithm, and
- $(\Sigma', \circledR, \geq_{\circledR})$ characterizes the focusing algorithm.

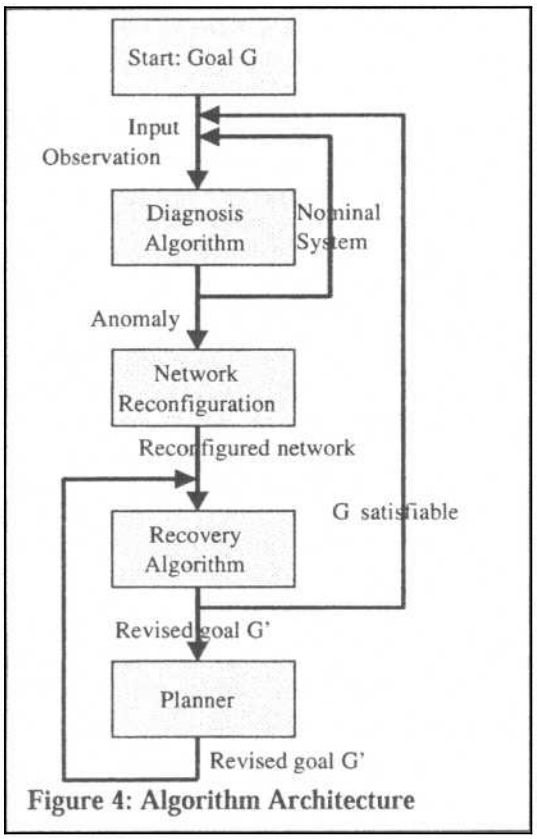

Figure 4: Algorithm Architecture

3.3 Diagnosis and Recovery Algorithm

The algorithm for this integrated approach is as shown in Figure 4. The steps are as follows:

1. Start the algorithm given a goal Θ and set of input observations.
2. Run the diagnosis algorithm.
   - If no anomaly is detected, return to 1.
   - If an anomaly is detected, identify the anomalous assumables $A_i \subseteq A$ and reconfigure the network.
3. Network reconfiguration.
   - Replace A_i by A_γ, such that A_γ is nominal, $A'=[(A \backslash A_i) \cup A_\gamma]$, and $A' \cup \Theta$ is not contradictory, i.e., we satisfy the goal Θ. We set the observables $O'=S \cup A_\gamma$.
   - Map control variables χ into assumables: $A'=\chi$.
4. Recovery Algorithm.
   - Run the recovery algorithm with inputs: (il) the dual network, and (ii) focusing based on the control algorithm. The output is a new control setting χ'.
     - If χ' meets the goal Θ goto 1.
     - If χ' does not meet goal Θ, send the model to the planner to revise the goal to Θ'.
5. Planner
   - Revise Θ to Θ' according to domain priorities.

3.4 Guarantees for Recovery Algorithm

This section summarizes the guarantees that we can make for the recovery algorithm. Note that these results are derived using a modified diagnosis inference procedure as described by Darwiche[1995,1998].

Lemma 1: Given a recovery model $\langle \Psi, \text{GOAL} \cup A^*, \text{Dx-Alg}, (\Sigma', \circledR, \geq_{\circledR}) \rangle$, the focusing algorithm is guaranteed to compute all sound, minimal recovery actions, if any exist.

Lemma 2:

1. Computing the set of recovery actions is NP-hard.
2. Using a causal network representation, in the worst case computing the set of recovery actions is linear

in the number of network nodes and exponential in the maximum graph width of the causal network.

One key approach that we can use is to pre-compute a representation R for the set of all recovery actions, cache that function, and then compute recovery actions on a case-by-case basis using the focusing algorithm [Darwiche,1998]. We call this diagnostic compilation. This means that the NP-hard task of generating R is performed only once, and particular recovery actions are computed efficiently as below:

Lemma 3: Given a representation R for the set of all recovery actions, computing a particular recovery action given an instantiation of assumables **A'** can be done in time $O(|R| \ |\circledR|)$, where $|\circledR|$ denotes the time for performing the ® operation followed by a minimization.

4 Diagnosis and Recovery for Rocket Engine Subsystem

In this section we show how we can apply our diagnosis and recovery approach to the rocket engine example.

4.1 Subsystem Models

Figure 2 shows the full causal network that corresponds to our example: assumables are depicted as boxes, unobservables as light-shaded ovals, and observables as dark-shaded ovals. For the purposes of clarity of explication, we focus on a subset of the model, the A-channel of the LOX system. Figure 5 shows the diagnostic causal network for this portion of the model. In this figure, the solid arcs represent the causal relationships for the system, or plant model; the dotted arcs represent the causal relationships for the control actions. For example, the setting for LOX-inlet-pressure is determined (through feedback control) by the Goals (encoded in the Thrust node) and the current value of the V_1-sensor.

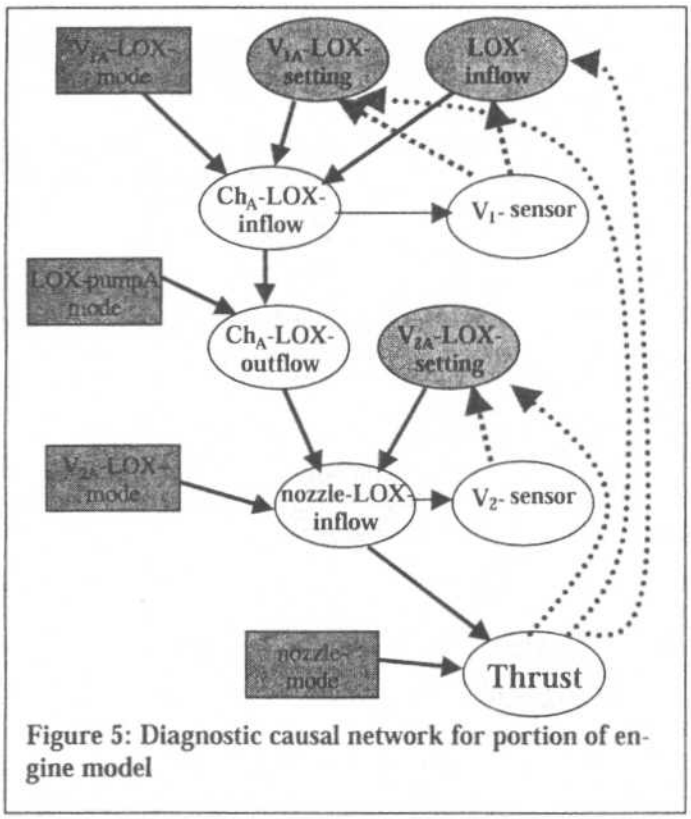

Figure 5: Diagnostic causal network for portion of engine model

Next, Figure 6 shows the recovery causal network for this portion of the model. Notice that in the recovery causal network, the assumables and control nodes have swapped roles.[1]

4.2 Diagnosis and Recovery Scenario

We now briefly describe a few diagnosis and recovery scenarios for the LOX lines of this rocket engine.

Cost-Based Recovery: The first set of scenarios use the cost-based control algebra. Table 6 summarizes these scenarios.

Scenario	1	2	3
Thrust Goal	≥75%	≥75%	≥75%
Obs. Thrust	50%	50%	25%
Control Settings	V_{1A}=open V_{2A}= open P_{in}=nom	V_{1A}= open V_{2A}= open P_{in} =nom	V_{1A}= open V_{2A}= open P_{in} =nom
Sensor Settings	S_{1A}=nom. S_{2A}=nom.	S_{1A}=nom. S_{2A}=low	S_{1A}=nom. S_{2A}=v-low
Min. Dx.	Nozzle= minor leak	$Pump_A$= low output	$Pump_A$= failed
Recovery	P_{in} = high	P_{in} = high	V_{1B}= open V_{2B}= open

Table 6: Diagnostic/Recovery Scenarios for Cost Algebra

In the first scenario, we want to achieve ≥75% Thrust using nominal control settings. Both sensors read nominal, yet the observed thrust is only 50%. The minimal diagnosis in this case is that there is a minor leak in the nozzle. If we run the recovery algorithm on the dual network, we compute a recovery action of setting the LOX inlet-pressure to be high.

In the second scenario, we want to achieve ≥75% Thrust using nominal control settings. Sensor S_{1A} reads nominal and S_{2A} reads low, and the observed thrust is only 50%. The minimal diagnosis in this case is that $Pump_A$ is operating at low output. If we run the recovery algorithm on the dual network, we compute a recovery action of setting the LOX inlet-pressure to be high. (Note that we could also have used the B-channel to obtain the desired thrust, but that would have led to a higher-cost recovery action.)

In the third scenario, we want to achieve ≥75% Thrust using nominal control settings. Sensor S_{1A} reads nominal and S_{2A} reads very-low, and the observed thrust is only 25%. The minimal diagnosis in this case is that $Pump_A$ has failed. If we run the recovery algorithm on

[1] For technical reasons, assumables cannot have parents, so we assign to the control nodes parents that play the role of assumables.

the dual network, we compute a recovery action of setting the V_{1B} and V_{2B} valves to nominal, thereby using the redundant B LOX channel.

Transition-Based Recovery: The second set of scenarios use the weighted transition-based control algebra. Table 7 summarizes these scenarios.

Scenario	1	2	3
Thrust Goal	≥75%	≥75%	≥75%
Obs. Thrust	50%	50%	25%
Control Settings	V_{1A}= open V_{2A}= open P_{in} =nom	V_{1A}= open V_{2A}= open P_{in} =nom	V_{1A}= open V_{2A}= open P_{in} =nom
Sensor Settings	S_{1A}=nom. S_{2A}=nom.	S_{1A}=nom. S_{2A}=low	S_{1A}=nom. S_{2A}=v-low
Min. Dx.	Nozzle= minor leak	$Pump_A$= low output	$Pump_A$= failed
Recovery	P_{in} = high	P_{in} = high	V_{1B}= 25% V_{2B}= 25% Inflo= high

Table 7: Diagnostic/Recovery Scenarios for Transition Algebra

The recovery actions for first and second scenarios are identical to those for the cost-based case. However, we will compute a different result for the third scenario. This is because we are constrained to a transition from the B-channel valves being closed to these valves being opened 25%, as shown in the valve transition chart in Figure 3. Note, however, that we can compensate for this partial opening by setting the LOX inlet-pressure to be high as well.

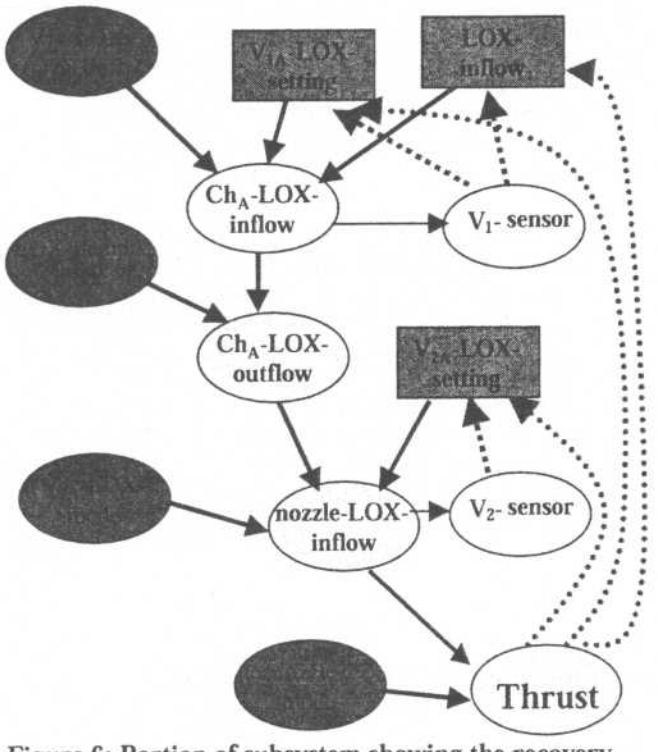

Figure 6: Portion of subsystem showing the recovery network

5 Related Work

This section compares and contrasts our approach to previous work on this and related topics.

This approach is most closely related to the Livingstone system described in [Williams and Nayak,1996]. This approach shares with Livingstone the computation of system state (or assumable binding) and recovery actions, called mode identification and mode reconfiguration, respectively, in [Williams and Nayak,1996]. Livingstone is defined using a transition logic, and its algorithms are related to, but different from the MBD algorithms. We show how we can use a standard MBD approach to achieve the same purposes as Livingstone. Another important distinction is that in our approach we can compile the model representation such that all diagnoses and recovery actions can be pre-computed and efficiently generated on-line in real time given sensor data. This compilation approach [Darwiche,1998] uses knowledge of the specified assumables and observables to pre-compute much of the diagnostic inference. The compiled diagnostic representation can be evaluated in (at worst) time linear in the size of the representation. Note, however, that the expression can be exponential in certain model parameters, such as graph width.[2]

There have been several other approaches to recovery, e.g., [Johnson,1996],[Rauch,1995]. The main difference between our approach and these other approaches is that

[2] For many applications, we have had significant success in reducing the size of the compiled expression by rewriting the model such that the resulting model parameters ensure a relatively small compiled representation.

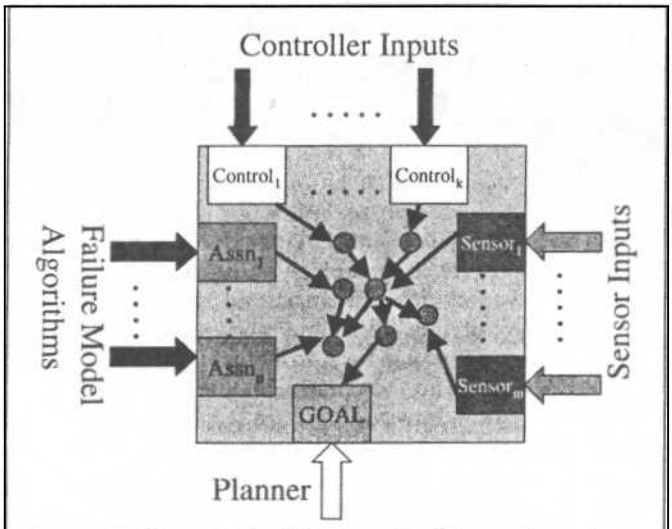

Figure 7: Generic Architecture for Diagnosis and Recovery System

our approach is based on MBD representations and algorithms, and aims to integrate both diagnosis and recovery.

6 Summary and Future Work

This document has proposed a novel approach to integrating diagnostic inference and recovery within a single formalism. We have shown how to extend a model-based diagnostic approach to incorporate control and recovery actions. We applied the modeling formalism and algorithms for a particular model-based approach, causal networks, to a simple rocket engine example. This new approach allows us to develop analytical results for recovery similar to those already developed for diagnostic inference.

More generally, we have designed our implementation such that it can be integrated with several other representations and algorithms using CORBA/COM protocols. Figure 7 shows how we can inter-operate with a variety of other inputs, from sensors, controller modules, planners and failure-model modules.

Although the control/reconfiguration actions currently generated by our approach are limited to the next-step actions, the "horizon" of the control actions can be extended by modeling the underlying systems as temporal causal networks [Darwiche and Provan,1996]. Based on a standard propositional temporal language, a temporal causal network models temporal relationships among system components in terms of variables with discrete time indices. We plan to further explore this approach in future work.

References

[Darwiche,1995] A. Darwiche. Model-based diagnosis using causal networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 211–217, 1995.

[Darwiche,1998] A. Darwiche. Model-based diagnosis using structured system descriptions. *Journal of AI Research*, 8:165-222, June 1998.

[Darwiche and Provan,1996] Darwiche, A. and Provan, G., "Exploiting system structure in model-based diagnosis of discrete-event systems." In Proceedings of International Workshop on Principles of Diagnosis, Canada, pp. 95-105. 1996.

[de Kleer and Williams, 1989] J. de Kleer and B. Williams. Diagnosis with Behavioral Modes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1324—1330, August 1989. Morgan-Kaufmann Publishers.

[Dressler and Struss,1996] O. Dressler and P. Struss. The Consistency-based Approach to Automated Diagnosis of Devices. In: Brewka, G. (ed.), *Principles of Knowledge Representation*, CSLI Publications, Stanford, ISBN 1-57586-057-0, pp. 267-311, 1996.

[Johnson, 1996] D. M. Johnson, 1996. Integrated modular avionics: a scheme for autonomous dynamic system reconfiguration. *Computer Systems Science and Engineering* v 11(3), pp. 125-133.

[Rauch, 1995] H. E. Rauch, 1995. Autonomous control reconfiguration. *IEEE Control Systems Magazine* v 15(6), pp. 37-48.

[Reiter, 1997] R. Reiter, 1997. A Theory of Diagnosis from First Principles. *Artificial Intelligence Journal*, v 32(1), pp. 57-95.

[Struss,1992] P. Struss. What's in SD? Towards a Theory of Modeling for Diagnosis. In: L. Console, W. Hamscher and J. deKleer (eds.), *Readings in Model-based Diagnosis*, Morgan-Kaufmann Publishers, 1992.

[Williams and Nayak,1996]. B. Williams and P. Nayak. A model-based Approach to Reactive Self-configuring Systems. In *Proceedings of the Conf. Of the American Assoc. for Artificial Intelligence (AAAI)*, pages 971–978, 1996.