# Improving Model-based Mode Estimation through Offline Compilation

Seung H. Chung, John M. Van Eepoel, Brian C. Williams

MIT Space Systems Laboratory
MIT Artificial Intelligence Laboratory
Cambridge, MA, 02139
{chung, vanny, williams}@mit.edu

**Keywords** Model-based, mode, estimation, dissents, conflicts, diagnosis, best-first, search, partial diagnoses.

## Abstract

Many recent and future space missions point to the need for increased autonomy in spacecraft with an emphasis on more capable fault diagnostic systems. The most widely used fault diagnostic systems are rule-based. Rule-based systems have quick response to events and clearly present to engineers the predefined reactions to events. These systems, however, require engineers to manually generate all necessary rules and these do not convey the assumed model the engineers used to generate the rules. Contrarily, model-based systems eliminate the need to manually generate the rules. Most model-based system such as GDE [3], Sherlock [4], and Livingstone [6], however, may not provide quick response and do not specify the rules for engineers to review and verify. Mini-ME addresses the issues of both rule-based and model-based approaches and provides an alternative solution to fault diagnosis. Mini-ME provides quick response by shifting computationally expensive tasks of model-based diagnosis offline. Additionally, it offers the capability to inspect and verify the rules.

## 1 Introduction

Recent failures in NASA's Mars exploration program point to the need for increased autonomy in spacecraft. Spacecraft must be designed with the capacity to monitor their own systems for unexpected occurrences, and to react in a timely fashion to such conditions at the executive layer, i.e. at the level of real-time commanding. The ability to accurately and rapidly determine the current state of the system is vital to the design of fault protection systems in autonomous spacecraft.

Many fault management systems are based on expert systems in which a rule-based diagnostic engine is used to detect faults. For example, the NEAR spacecraft used such a system for limited autonomous operations. This type of system's capability is limited to the rules enumerated in the database. To create these rules, engineers must reason through system wide interactions, consequently, the set of rules is limited by the faults that engineers can recognize. This lack of robustness can be detrimental to the spacecraft. Components may interact other than expected, and a rule-based system cannot account for this. Should such a fault occur at a critical mission point, such as orbital insertion, the rule-based engine cannot react, resulting in the loss of the mission.

In contrast, model-based fault protection systems eliminate the need to enumerate all rules by reasoning on the common-sense model of a system. Model-based fault protection systems such as Sherlock [4] and Livingstone [6], however, are limited by the uncertainty in the time required to perform online automated reasoning. Online deduction algorithms, such as those used in the Livingstone model-based fault protection system, are known to have worst-case performance exponential in time. The issue of run-time uncertainty presents a challenge to the onboard implementation of model-based protection capabilities, a challenge certain to be highly exacerbated as systems become more complex and the modeling languages used in such systems become more expressive.

As a solution to this problem, a Miniature Mode Estimation system (Mini-ME) is introduced. Mini-ME estimates the current mode of the spacecraft and detects any failures through model-based reasoning. The following sections detail the Mini-ME system beginning with the modeling framework, followed by the offline and online tasks of fault diagnosis. Then, an analysis is presented using the NEAR spacecraft and how Mini-ME is capable of diagnosing the modes of the power sub-system.

## 2 Mini-ME

Mini-ME differs from previous model-based fault monitoring systems by guaranteeing run time

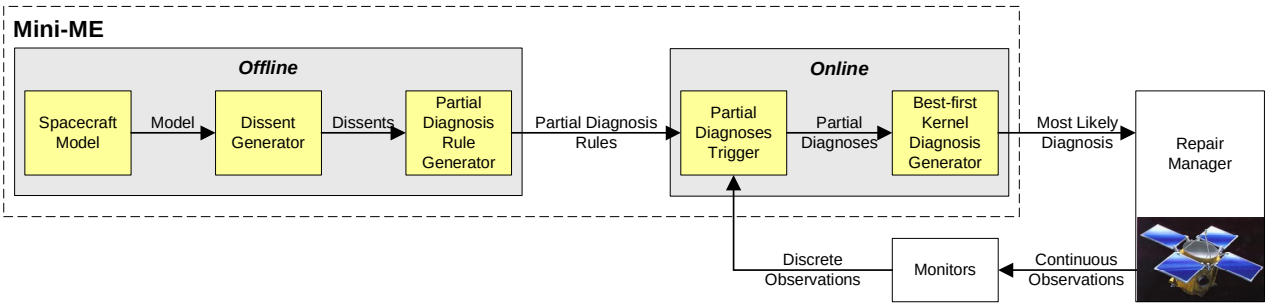


Figure 1: Mini-ME Architecture

performance. Through model compilation and offline deduction, Mini-ME combines the benefits of the rule-based system's real time performance guarantees and the model-based fault protection system's capability to reason on models.

The architecture of the Mini-ME system is shown in Figure 1. The engineer specifies commonsense models of the spacecraft containing the operational and fault modes of each component and its associated behavior. The model is then compiled into a set of rules, called dissents. Dissents represent a mapping from observations of the system to a set of possible diagnoses of the component modes. This mapping is computed offline and thus does not factor in to the real time performance of the system.

These partial diagnosis rules are then fed into the partial diagnosis trigger that generates a set of partial diagnoses of the components based on the observations in the current time step. These observations come from the spacecraft, but are discretized through the use of monitors. For example, voltage is read as a continuous value, but for the spacecraft to operate properly, this voltage need only be within a certain range, such as 24 to 32 volts. This range is discretized as nominal, and anything else is either low or high as appropriate.

These partial diagnoses are then combined in the best-first kernel generation step to give the most likely diagnosis of the system. This diagnosis is the output of Mini-ME and the input to the repair manager, which will then use this diagnosis to take an appropriate recovery action to repair components.

## 2.1 Example System

The diagnostic ability of Mini-ME will be demonstrated in the following sections using a simplified schematic of a monopropellant propulsion system used for attitude control in the NEAR spacecraft, shown in Figure 2.

The propulsion system comprises two overall sub-systems, the tank of hydrazine and its associated pressure transducer, and the hydrazine thruster made up of the solenoid valve, catalyst bed and physical thruster. An inertial sensor is included in the system for thrust observation.

The pressure transducer indicates the pressure that is present in pipe 1, which is downstream of the tank. As long as the pressure transducer is operating nominally, it will indicate that the pressure in pipe 1 is the same as the tank pressure, discretized as either nominal or low.

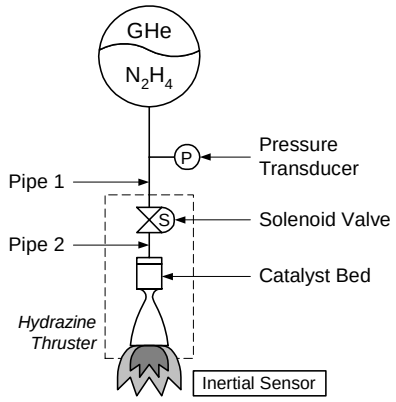


Figure 2: Monopropellant Propulsion System Schematic

The hydrazine thruster is made up of two main components, the solenoid valve and the catalyst bed. The solenoid valve controls the hydrazine flow into the catalyst bed. This is accomplished by applying an electric current to the valve to open it, otherwise it will remain closed. Downstream of the solenoid valve is the catalyst bed, which is needed for combustion. Over time, catalyst can be lost through various mechanisms, such as pieces breaking off due to temperature variations. This will cause a reduction in thrust from the hydrazine thruster, causing the inertial sensor to observe that the thrust is off. In the case that the ACS operates nominally, the inertial sensor's reading will be discretized as on.

## 2.2 Modeling

System modeling is key to the utility of the Mini-ME diagnostic tool. The utility of a modeling language allows specification of commonsense models that are also reusable and compositional. The reusability comes from the general formulation of a model. The compositional aspect of the model exists at the system

level by specifying components of a system, but also at the component level by specifying the different modes of each component. These benefits inherent in a model-based approach give Mini-ME the ability to perform its diagnosis. The following demonstrates the modeling approach, where sensor information is identified, components operational and fault modes are determined, and their associated models are developed.

The propulsion system detailed above has two sensors. The pressure transducer measures the pressure in pipe 1 and the inertial sensor measures the thrust. There are also points in the system where information cannot be observed, such as the pressure in the tank and pipe 2. The discretized observations are detailed below.

**Observable Variables**

Pressure in Pipe 1 - P1: {*nom*, *low*}
(P1 = *nom*) ⊗ (P1 = *low*)

Engine Thrust - T: {*on*, *off*}
(T = *on*) ⊗ (T = *off*)

**Unobservable Variables**

Tank Pressure - TP: {*nom*, *low*}
(TP = *nom*) ⊗ (TP = *low*)

Pressure in Pipe 2 - P2: {*nom*, *low*}
(P2 = *nom*) ⊗ (P2 = *low*)

The models of this system are detailed below where "S" represents the pressure transducer sensor, "V" represents the solenoid valve and "C" represents the catalyst bed. Each model has certain operational and failure modes with associated probabilities. These probabilities help guide the online components of Mini-ME. In real systems, some components have a higher likelihood of failing than others. In this system, it is more common for electrical sensors to become faulty and give incorrect data, than for a component to enter a failure state. Due to this fact, the probabilities of failure of the pressure transducer are higher than the probability of failure of any single component. These probabilities are presented in Table 1.

With this in mind, the modes of each component can now be defined. In the case of the pressure transducer (S), there are four modes in this variable's domain. The transducer can be good (G), stuck high (SH), stuck low (SL) or in an unknown (U) mode. The logic that specifies the models of these modes is shown below.

**Pressure Transducer Model**

S: {G, SH, SL, U}

G(S) ⊗ SH(S)⊗ SL(S) ⊗ U(S)

G(S) ⇒
((TP = *nom*) ⇔ (P1 = *nom*)) ∧ ((TP = *low*) ⇔ (P1 = *low*))

SH(S) ⇒(P1 = *nom*)

SL(S) ⇒(P1 = *low*)

The pressure transducer can only be in one mode at a given time. This constraint is encoded using an exclusive or (⊗). To satisfy this proposition, only one of the values from its domain can be chosen.

Being in the good mode, G(S) of the pressure transducer then means that the tank pressure can only be at the same pressure as the pressure in pipe 1. For the case of the SH(S) and SL(S) modes, the pressure transducer is faulty and only returning high or low values, respectively. These values are returned no matter the actual value of the pressure in the system. The unknown mode has no associated model, and is not shown for any component. This mode is included in the component model to cover unanticipated behaviors, which by definition cannot have an associated model.

The component mode probabilities of this sensor must satisfy the aforementioned constraint between sensors and components. This gives rise to the SH(S) and SL(S) fault modes having a "less likely" probability (see Table 1) rather than an "unlikely" probability. The sensor probability constraint dictates that the U(S) mode probability also be higher than those of the component mode.

Table 1. Component Mode Probabilities

| | **Components** | | |
|---|---|---|---|
| **Mode Probabilities** | Pressure Transducer | Solenoid Valve | Catalyst Bed |
| | p(G) = very likely | P(C) = very likely | p(G) = very likely |
| | p(SH) = less likely | p(O) = likely | p(B) = unlikely |
| | p(SL) = less likely | p(U) = very rare | p(U) = very rare |
| | p(U) = rare | -- | -- |

very rare = 0.001 rare = 0.002 unlikely = 0.009
less likely = 0.019 likely = 0.089 very likely ≥ 0.9

**Solenoid Valve Model**

V: {O, C, U}

O(V) ⊗ C(V) ⊗ U(V)

O(V) ⇒
((P1 = *nom*) ⇔ (P2 = *nom*)) ∧ ((P1 = *low*) ⇔ (P2 = *low*))

C(V) ⇒(P2 = *low*)

The solenoid valve has modes corresponding to its physical state of being open, O(V), or closed, C(V). The only failure mode corresponds to failure in some unknown way, such as breaking to remain closed, remain open or some other way. For an open valve, this implies that the pressure in pipe 1 can only be the same as the pressure in pipe 2. For a closed valve, it can only be the case that the pressure in pipe 2 is low. The component mode probabilities of the solenoid differ by an order of magnitude due to its actuation. It is more likely that the valve will be closed because it requires electrical actuation to open it, and this is less likely to occur.

**Catalyst Bed Model**

C: {G, B, U}

G(C) ⊗ B(C) ⊗ U(C)

$G(C) \Rightarrow$
$((P2 = nom) \Leftrightarrow (T = on)) \wedge ((P2 = low) \Leftrightarrow (T = off))$
$B(C) \Rightarrow (T = off)$

The catalyst bed remains functional as long as there is catalyst remaining to react with the incoming hydrazine and that the temperature is appropriate. Then, the good (G) mode corresponds to producing thrust only when the pressure in pipe 2 is nominal, and producing no thrust only when the pressure is low. Many factors can create a broken catalyst bed, and the end result is that the thrust is off, meaning that very low values of thrust are discretized to 'off'.

Developing these types of models represents a paradigm shift from rule based systems where previously the engineer not only had to determine the failure modes of components, but how a failure would impact the overall system. Now, the engineer need only determine the underlying behavior of components, or the models of components.

## 2.3 Generating Dissents

The first offline step in Mini-ME is compiling a model into *dissents*. Each dissent may be viewed as a rule that identifies a conflict between a subset of the observations and a subset of component modes. For example, one of the dissents of the aforementioned propulsion system model is:

$$(P1 = nom) \wedge (T = off) \Rightarrow \neg(G(S) \wedge O(V) \wedge G(C))$$

which says that if the pressure in Pipe 1 is nominal and the thrust is off, then it cannot be the case that the pressure transducer is good, the latch valve is open, and the catalyst bed is good". Formally, dissents are a subset of so-called prime implicates [1] whose clauses only include the observable and mode variables. The complete set of dissents for the aforementioned propulsion system model is:

1. $(P1 = nom) \wedge (T = off) \Rightarrow \neg(G(S) \wedge O(V) \wedge G(C))$
2. $(P1 = nom) \Rightarrow \neg SL(S)$
3. $(P1 = low) \wedge (T = on) \Rightarrow \neg(G(S) \wedge O(V) \wedge G(C))$
4. . $(P1 = low) \Rightarrow \neg SH(S)$
5. $(T = on) \Rightarrow \neg C(V)$
6. $(T = on) \Rightarrow \neg B(C)$

The dissents are written in the form of observation implies conflicting mode(s) of component(s).

**Enumerating Dissents**

Several methods of generating dissents exist, including resolution [2], enumeration [7], and multi-resolution [1]. The current method used is an enumeration method. In this method, all possible dissent forms, i.e. all combinations of observation(s) imply inconsistent component mode(s), are systematically generated from smallest to largest clauses. A satisfiability engine is used to check if each of these dissent forms is entailed by the model. When, a dissent form is confirmed as dissent. The knowledge of the found dissents is used when generating more dissent forms to assure that no superset of the found dissent is generated. This assures that dissents are indeed subset of prime implicates.

**Benefits of Compiling Models into Dissents**

The advantage of compiling a model into a set of dissents is that all irrelevant information in the model is removed and only the information necessary for diagnosis is kept. Additionally, dissents are intuitive to engineers; thus, engineers can read through the dissents to verify the correctness and completeness of the model. In a rule-based fault diagnostic system, engineers would manually generate this type of knowledge, but generating complete and correct rules is difficult. In Mini-ME, however, the dissents are automatically generated and are guaranteed to be complete and correct for the given model.

## 2.4 Mapping Dissents to Diagnostic Rules

Mini-ME rewrites the dissents in a form that is more useful for diagnosis. In diagnosis the objective is to determine the modes of components that agree with the current observations. A dissent lists a set of component modes that are mutually inconsistent for given observations. To use dissent for diagnosis, a dissent is rewritten as a set of probable component modes that corresponds to the observations, i.e. the diagnosis for given observations. For example, consider the first dissent from the previous section. Intuitively, for the given observation, it is not the case that the sensor is good, the valve is open, and the catalyst bed is good. To remove this conflict the sensor must be either stuck high, stuck low, or unknown, or the valve must be closed or unknown, or the catalyst bed must be broken or unknown. The list of dissents from the previous section rewritten in the form "observation implies one of a set of component modes" for diagnostic use is as follows:

1. $(P1 = nom) \wedge (T = off) \Rightarrow SH(S) \vee SL(S) \vee U(S) \vee C(V) \vee U(V) \vee B(C) \vee U(C)$
2. $(P1 = nom) \Rightarrow G(S) \vee SH(S) \vee U(S)$
3. $(P1 = low) \wedge (T = on) \Rightarrow SH(S) \vee SL(S) \vee U(S) \vee C(V) \vee U(V) \vee B(C) \vee U(C)$
4. $(P1 = low) \Rightarrow G(S) \vee SL(S) \vee U(S)$
5. $(T = on) \Rightarrow O(V) \vee U(V)$
6. $(T = on) \Rightarrow G(C) \vee U(C)$

In essence, this new form is the diagnosis that resolves a conflict of some specified observation(s). For a set of observations, however, multiple conflicts can exist. For example, consider a set of observations $\{(P1 = low), (T = on)\}$. The dissents list four conflicts, $\neg(G(S) \wedge O(V) \wedge G(C))$, $\neg SH(S)$, $\neg C(V)$, and $\neg B(C)$, that are associated with the set of observations. Accordingly, four diagnoses, $SH(S) \vee SL(S) \vee U(S) \vee C(V) \vee U(V) \vee B(C) \vee U(C)$, $G(S) \vee SL(S) \vee U(S)$,

$O(V) \vee U(V)$, and $G(C) \vee U(C)$ resolve the conflicts respectively, but none of the three diagnoses resolves all three conflicts. Since each of the rewritten dissents is a diagnosis of a conflict of possibly many, these rewritten dissents are called *partial* diagnosis rules. Formally, mapping dissents to partial diagnoses simply corresponds to rewriting the negation of conjunction of component modes (conflicts) into a disjunction of component modes (partial diagnosis).

## 2.5 Triggering Rules

The first online step in Mini-ME is triggering rules. As shown in Figure 3, Mini-ME's rule trigger requires two inputs, current observations and the partial diagnosis rule database. If the observation in a partial diagnosis rule corresponds to the current observation, then the modes the rule implies are the relevant partial diagnosis. Consider the partial diagnosis rules listed in the previous section. If the current observation is that the pressure in pipe 1 is nominal and the thrust is off, the rules that contain relevant partial diagnosis are rules 1 and 2, and the corresponding partial diagnoses in set notation are:

{SH(S), SL(S), U(S), C(V), U(V), B(C), U(C)}
{G(S), SH(S), U(S)}

The partial diagnoses triggered by each of the four sets of possible observations are:

- (P1 = *nom*) $\wedge$ (T = *off*):
  {SH(S), SL(S), U(S), C(V), U(V), B(C), U(C)}
  {G(S), SH(S), U(S)}
- (P1 = *nom*) $\wedge$ (T = *on*)
  {G(S), SH(S), U(S)}
  {O(V), U(V)}
  {G(C), U(C)}
- (P1 = *low*) $\wedge$ (T = *off*)
  {G(S), SL(S), U(S)}
- (P1 = *low*) $\wedge$ (T = *on*)
  {SH(S), SL(S), U(S), C(V), U(V), B(C), U(C)}
  {G(S), SL(S), U(S)}
  {O(V), U(V)}
  {G(C), U(C)}

**Difference between Sherlock or Livingstone and Mini-ME**

GDE [3], Sherlock [4], Livingstone [6], and many alike use conflicts to generate fault diagnosis as Mini-ME does. The difference between these and Mini-ME is that while GDE, Sherlock and Livingstone use online satisfiability engines to generate the conflicts corresponding to the current observation, Mini-ME performs a simple partial diagnosis rule lookup to search for partial diagnoses that correspond to the conflicts. The advantage in Mini-ME is that the NP-complete satisfiability problem is removed from online computation. Instead of requiring possibly an exponential satisfiability search, Mini-ME generates partial diagnoses in time that is linear in the number of rules associated with a set of observations.

## 2.6 Generating Kernel Diagnosis

First introduced in GDE [3], the final step in Mini-ME is to generate the most likely kernel diagnosis from the partial diagnoses. A kernel diagnosis is a minimal set of component modes that resolves a full set of conflicts. The input to Kernel Diagnosis Generation is a set of partial diagnoses generated from the Rule Trigger. Kernel Diagnosis Generation step generates a single most likely kernel diagnosis that concurs with all partial diagnoses. The following sections provide an overview of generating kernel diagnosis introduced in GDE.

**Kernel Diagnoses**

To generate a kernel diagnosis is to resolve all conflicts. Since partial diagnoses resolve the corresponding conflicts, some combinations all triggered partial diagnoses may be the kernel diagnoses. The right combinations are generated by choosing one component mode from each partial diagnosis while assuring that this choice does not disagree with any other chosen component modes, i.e. no component is assigned multiple modes. For example, if O(V) is chosen from one partial diagnosis, then C(V) or U(V) cannot be chosen from other partial diagnoses. This process of generating the kernel diagnosis is in essence generating the minimal set covering of the input diagnoses.

Consider again the partial diagnoses generated when the system observes that the pressure in pipe 1 is nominal and the thrust is off. The partial diagnoses generated by the conflict generation step are:

{SH(S), SL(S), U(S), C(V), U(V), B(C), U(C)}
{G(S), SH(S), U(S)}

If SL(S) were chosen from the first partial diagnosis, no component mode can be chosen from the second partial diagnosis without assigning the sensor a second mode. Instead, SH(S) is chosen from the first partial diagnosis, then the only possible choice from the second partial diagnosis is SH(S). Hence, a kernel diagnosis is SH(S). A third option is to choose C(V) from the first partial diagnosis. Then, any component mode can be chosen from the second partial diagnosis to generate a kernel diagnosis. The complete list of kernel diagnoses is:

- [SH(S)]
- [U(S)]
- [G(S), C(V)]
- [G(S), U(V)]
- [G(S), B(C)]
- [G(S), U(C)]

Note that although [SH(S), C(V)] and many others are also valid coverings, they are not minimal, and hence are not listed as kernel diagnoses.

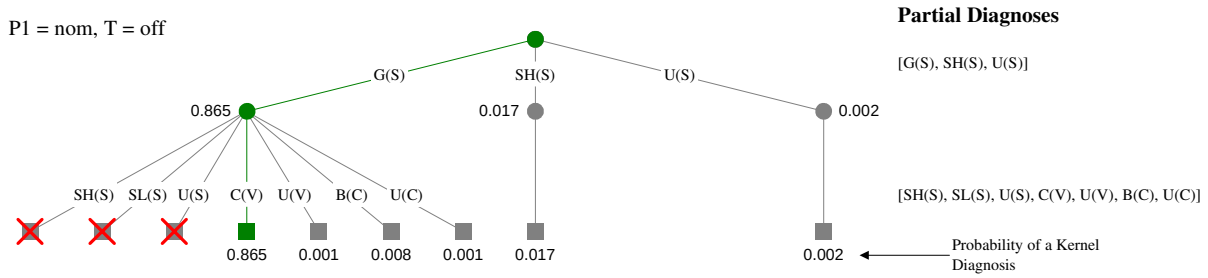


Figure 3: A* search tree for kernel diagnosis

### Most Likely Kernel Diagnosis

Mini-ME ranks likelihood of kernel diagnoses based on probabilities. To compute kernel diagnosis probability, each component mode is assumed probabilistically independent. The probability of a kernel diagnosis is calculated by taking the product of the component mode probabilities. Kernel diagnosis, however, is not a complete diagnosis. Kernel diagnosis specifies only a subset of component modes such that all conflicts are resolved. For the components that are not associated to any conflicts, their modes are free to be assigned to any of their modes. A kernel diagnosis is extended to a full diagnosis by assigning the most likely modes to the unspecified components. The probability of the kernel diagnosis is defined as the probability of the full diagnosis that is an extension to the kernel diagnosis:

$$p(KD_i) = p\left(ED_i\right) = \prod_{m \in ED_i} p\left(m\right) \qquad (1)$$

where the $KD_i$ is a kernel diagnosis, $ED_i$ is the most likely extension of the kernel diagnosis, and $m$ is a component mode in $ED$. For example, the probability of the kernel diagnosis [SH(S)] is equal to its most likely full diagnosis extension [SH(S), C(V), G(C)]. Therefore, the probability of the kernel diagnosis is equal to the product of the probabilities of SH(S), C(V), and G(C). The probabilities of each kernel diagnosis for the example with observations pressure in pipe 1 is nominal and the thrust is off are:

- $p([\text{SH(S)}]) = p(\text{SH(S)}) \cdot p(\text{C(V)}) \cdot p(\text{G(C)}) = 0.017$
- $p([\text{U(S)}]) = p(\text{U(S)}) \cdot p(\text{C(V)}) \cdot p(\text{G(C)}) = 0.002$
- $p([\text{G(S), C(V)}]) = p(\text{G(S)}) \cdot p(\text{C(V)}) \cdot p(\text{G(C)}) = 0.865$
- $p([\text{G(S), U(V)}]) = p(\text{G(S)}) \cdot p(\text{U(V)}) \cdot p(\text{G(C)}) = 0.001$
- $p([\text{G(S), B(C)}]) = p(\text{G(S)}) \cdot p(\text{C(V)}) \cdot p(\text{B(C)}) = 0.008$
- $p([\text{G(S), U(C)}]) = p(\text{G(S)}) \cdot p(\text{C(V)}) \cdot p(\text{U(C)}) = 0.001$

The most likely kernel diagnosis is [G(S), C(V)] which says that the pressure transducer is good and the solenoid valve is closed. Its most likely extended diagnosis is pressure transducer is good, solenoid valve is closed, and the catalyst bed is good.

### Generating Kernel Diagnosis using Best-first Search

The objective is to find the minimal set covering in the best-first order such that the search time is minimized. Similar to Livingstone, minimal set covering is accomplished through a conflict-directed best first search. The key to formulate minimal set covering as a tree search problem and search the tree in the best-first order.

*Generating Minimal Set Covering Tree*

Minimal set covering is achieved through constructing a tree of partial diagnoses. The tree's branches correspond to component modes of the partial diagnoses. The path from the root to a leaf represents minimal set covering of partial diagnoses, i.e. a kernel diagnosis. For clarification, consider the previous propulsion system example with a set of observations {(P1 = *nom*), (T = *off*)}. Figure 3 illustrates the tree generated for the kernel diagnoses of the set of observations. Again, the partial diagnoses for this set of observations are:

{SH(S), SL(S), U(S), C(V), U(V), B(C), U(C)}
{G(S), SH(S), U(S)}

Starting From the root, branches corresponding to each component modes in a set of partial diagnosis are expanded. In Figure 3, the first level branches correspond to each component modes of the partial diagnosis {G(S), SH(S), U(S)}. Then from each of the leaf nodes next partial diagnosis' component modes are expanded into branches. This process is repeated until all partial diagnoses have been covered. In Figure 3, the second level branches correspond to each component modes of the partial diagnosis {SH(S), SL(S), U(S), C(V), U(V), B(C), U(C)}. Note that the three leftmost branches on the second level are crossed off. This corresponds to the case in which two different modes are assigned to the sensor. In general, if a component mode is specified in the parent branches, no children branches with the same component but with a different mode are expanded. Also, in Figure 3, the middle and the rightmost branches of the first level, SH(S) and U(S), are not expanded further on the second level. Since both SH(S) and U(S) are a member of the second level branches, SH(S) and U(S) alone are the minimal set coverings, so no other branches need to be expanded. In general, if parent branch's component mode is repeated in the child branch, no sibling branches are expended.

*Searching the Tree in Best-first Order*

A* [5] search algorithm is used to generate kernel diagnoses from most likely to least likely. A* search,

however, assumes additive property of the edge weights to calculate the total cost while the probabilities of independent events are multiplicative as shown in Eq (1). The multiplicative probability is reframed as an additive edge weight by taking the logarithm of the probability and multiplying by -1:

$$w(m) = -\log\big(p(m)\big) \tag{2}$$

where $w(m)$ is the edge weight of the component mode $m$ that the branch represents. The sole purpose of multiplying by –1 is to maintain positive cost. Then, the cost function $f(n)$ is:

$$f(n) = \sum_{l \in SC(n)} w(l) + \sum_{m \in USC(n)} w(m) \tag{3}$$

were $n$ is a node in the tree, $SC(n)$ are the specified component modes on the path from the root to the node $n$ and $USC(n)$ are the most likely modes of components that are unspecified on the path from the root to the node $n$. Formally, the first part is the path cost from the root to the node $n$ and the second part is the estimated cost from the node $n$ to the goal, a full diagnosis.

Note that when a component mode is repeated more than once, the cost of that component mode is added only once or any repeating component mode's edge weight is assumed to be zero, and equivalently, the probability is considered to be one. This is necessary since if a component mode is repeated more than once this should not decrease the probability of the diagnosis rather it should remain the same. This is equivalent to calculating $p(m|m)$ which is equal to one. Furthermore, no other sibling branches at this level should be searched since no other sibling branches can have higher edge weight than one.

Then, the path corresponding to the minimal cost corresponds to the most likely minimal set covering, i.e. most likely kernel diagnosis. This best-first kernel diagnosis search method is guaranteed to be complete and optimal as A* search algorithm is known to be complete and optimal.

# 3 Rule System Analysis

A comparison to a real system is the best validation for the Mini-ME fault diagnosis tool. For verification, a NEAR-like power system and its associated rules were analyzed to develop appropriate Mini-ME models to obtain diagnoses of particular faults. The NEAR-like power storage subsystem schematic is shown in Figure 4. The associated rules of the system are shown in Table 2.

These rules have several characteristics relating to Mini-ME, the first being the dependency on time. In all of the rules, the observation must be made for a certain length of time before it is triggered. This dependency is moved outside of Mini-ME through the use of the monitors. Monitors can be designed with a counter that is incremented when an observation falls in a certain range, such as if the charger current exceeds 0.8 A. Only when the counter reaches a certain value, corresponding to 10 seconds for rule 3, then would the monitor send the observation that the charger current is "high". This use of discretization allows the modeling to be more intuitive and understandable as the model is now specified in a more qualitative way.

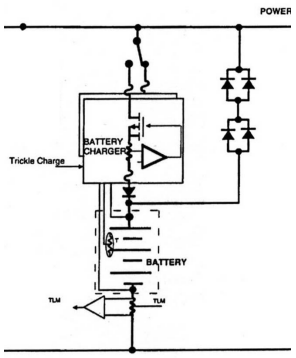


Figure 4 - NEAR-like Power Storage Sub-system

Table 2. NEAR-like Power Storage System Rules

| No. | Symptom | Recovery Action |
|---|---|---|
| 1 | (Battery Current > 0.6A) for 60 sec | Turn off the charger |
| 2 | (Redundant battery charger is ON) for 5 sec | stop rules 2 and 3 |
| 3 | (Charger current > 0.8 A) for 10 sec | Switch to the redundant charger, and disengage the primary. |
| 4 | (Charger current > 0.07A) and (Bus Voltage > 24 V) for 10 sec | Same recovery as rule 2. |
| 5 | (Battery Temp > 30 C) for 1 hour | Switch to the redundant charger and turn its trickle charge on |

A second characteristic of a NEAR-like rule is that different symptoms can lead to the same recovery action, such as the conditions for switching to the redundant charger (rules 3 and 4). These types of rule combinations may have the same resulting action, but lead to a different state for the component. Hidden in these rules then is the state of the power system that the engineer had to determine. For instance, in the case of rule 3, this would mean that the charger has become broken in some way, thus identifying the state, and the model of this mode would come from these symptoms.

Rules are easier to assess if they are atomic, which is done by making the state of the system explicit. Mini-ME achieves this by separating state determination from a recovery action through the use of models. Models are more intuitive and compositional, making the mapping from symptoms to system states to recovery actions easier to specify. A systems engineer can reason through the model of a component as it is based on the physical behavior of the component, as opposed to the

interactions of components. The latter approach has great potential for error and overlooked interactions.

The Mini-ME system is able to determine the intermediate states for each fault in the power system rules given the appropriate symptoms specified in each rule and a model for the system. As an example, the model for a battery charger will contain information about the values of its inputs such as its current reading, allowing for diagnosis of rule 3. Demonstration of this similarity between NEAR-like autonomy rules and rules specified by the Mini-ME tool is crucial in demonstrating parity of the two systems, but a full explanation of this analysis is deferred due to the limited scope of this paper.

Mini-ME has also been used as a tool to understand model compilation techniques and its applications. Other such applications that use this technique include a mode estimation capability for the Reactive Model-based Programming Language (RMPL) [8], and a reactive planning system, Burton [7].

# 4 Conclusions

Fault protection in spacecraft is a must as missions venture further into space and space systems increase in complexity. The necessity of a system that can perform this fault diagnosis in real time is then a key component. The Mini-ME fault protection system has been shown to meet this goal without any loss of information from a rule-based system.

The utilization of system models in Mini-ME allows it to perform diagnosis of components. A model-based approach has many benefits including reusability, compositionality and specification of intuitive models. The use of these models to perform reasoning and deduction has been shifted to an offline operation, an approach that differs from previous systems such as Sherlock and Livingstone. This offline compilation of the models to rules, called dissents, allows Mini-ME to perform fast diagnosis of faults online. Using these models and observations from the system, Mini-ME generates a diagnosis of the system's components using a best first search to generate the most likely diagnosis.

This diagnosis gives the state of the system, which is not available in a rule-based system. In rule-based systems, the mapping from symptoms to recovery action is apparent, but not the mapping from symptoms to the system state. Making this step explicit leads to rules that are easier to analyze for completeness, and a rule set smaller in size. In the case of the example system in section 2, it requires only 6 dissents to represent the faults, whereas a rule-based system would require 32 rules to represent all of the possible faults. These characteristics lead to more reliable fault protection as it makes the process of rule generation modular by using models of the system, monitors that discretize observations and repair actions based on the diagnoses, all of which are designed by the engineer in a clear manner.

A key benefit of the Mini-ME system and the use of associated repair manager, aside from the model-based approach, is that they give the spacecraft the ability to remain operational in the face of component failures. This ability is crucial as space exploration expands. The same individuals who designed the spacecraft may not be around when it lands, which necessitates fault diagnosis ability.

# Acknowledgements

We would like to thank Kenneth Heeres and Dave Watson of the Johns Hopkins Applied Physics Laboratory, and Michel Ingham, Samidh Chakrabati, Alvar Otero, and Michael Hofbaur of MIT. This research was supported in part by the DARPA MO-BIES program under contract F33615-00-C-1702 and by NASA's Cross Enterprise Technology Development program under contract NAG2-1466.

# References

[1] P. Chatalic and L. Simon, "Multi-Resolution on Compressed Sets of Clauses," In Proc. of 12th International Conference on Tools with Artificial Intelligence, (ICTAI-2000), Vancouver, 2000.

[2] J. de Kleer, "An improved incremental algorithm for generating prime implicates," In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), 780-785, 1992

[3] J. de Kleer and B. Williams. "Diagnosing Multiple Faults," Artificial Intelligenc, 32:100-117, 1987.

[4] J. de Kleer and B. Williams, "Diagnosis with Behavioral Modes," In Proceedings of the 11th International Joint Conference on AI (IJCAI-89), Detroit, MI, 1989.

[5] S. Russel and P. Norvig, *Artificial Intelligence: Amodern Approach*, Prentice Hall, 1995, New Jersey, Chap. 4 pp. 96-101.

[6] B. Williams and P. Nayak, "A Model-based Approach to Reactive Self-Configuring Systems," In Proceedings of AAAI-98, 971-978, 1996.

[7] B. Williams and P. Nayak. 1997. "A Reactive Planner for a Model-based Executive." In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-97).

[8] B. Williams, S. Chung, and V. Gupta. Mode Estimation of Model-based Programs: "Monitoring Systems with Complex Behavior," To appear in *Proceedings of the International Joint Conference on Artificial Intelligence*, Seattle, WA. 2001.