Chapter 8
Diagnosis of Hybrid Systems Using Structural Model Decomposition

Matthew J. Daigle, Anibal Bregon, and Indranil Roychoudhury

8.1 Introduction

Automated fault diagnosis is critical for complete system autonomy. In order for engineering systems to function in the real world, including extreme environments, the ability to self-diagnose faults and failures and then mitigate them by control or repair actions is crucial. Many engineering systems are *hybrid* in nature, i.e., they exhibit both continuous and discrete behaviors. The combination of continuous and discrete dynamics makes the problem of robust and efficient fault diagnosis significantly more challenging.

In a hybrid system, the system behavior is defined by a set of discrete *modes*. In each of these modes, a different set of continuous dynamics governs the system behavior. Discrete dynamics define how the system transitions from one mode to another. For example, consider an electric circuit with ten switching elements. If each switch can be in one of two states (the *on* state or the *off* state), then such a system has 2^{10} possible system-level modes. Therefore, a diagnosis algorithm, in general, must consider all possible modes of such a system.

M. J. Daigle (✉)
NIO USA, Inc., San Jose, CA, USA
e-mail: matthew.daigle@nio.io

A. Bregon
Departamento de Informática, Universidad de Valladolid, Valladolid, Spain
e-mail: anibal@infor.uva.es

I. Roychoudhury
Stinger Ghaffarian Technologies, Inc., NASA Ames Research Center, Moffett Field, Mountain View, CA, USA
e-mail: indranil.roychoudhury@nasa.gov

M. Sayed-Mouchaweh (ed.), *Fault Diagnosis of Hybrid Dynamic and Complex Systems*, https://doi.org/10.1007/978-3-319-74014-0_8

Further, faults may manifest as direct changes in system parameters, called *parametric faults*, or as changes in the system mode, called *discrete faults*. So a diagnosis system must reason over different types of faults and possible mode transitions during the fault isolation process. The effects of a fault are also mode-dependent, and observation delays (e.g., due to delays in signal filtering within a fault detection algorithm, or communication delays) may cause the observed effects to be inconsistent with the current mode of the system, but consistent with a previous mode. All of these complications can significantly complicate the reasoning process [1].

Researchers have been intrigued by the fault diagnosis problem for hybrid systems for many years, and many different proposals for hybrid systems diagnosis exist in the literature. During the last decade or so, modeling and diagnosis for hybrid systems has been an important topic of research from both Systems Dynamics and Control Engineering (FDI) and the Artificial Intelligence Diagnosis (DX) communities. In the FDI community, several hybrid system diagnosis approaches have been developed, where parameterized ARRs are used [2, 3]. However, such approaches are not suitable for systems with high nonlinearities or a large set of modes. In the DX community, some approaches have used hybrid automata to model the complete set of modes and transitions between them. In those cases, diagnosis is viewed as a hybrid system state estimation problem, and approached through probabilistic [4, 5] or set-theoric approaches [6]. Another solution has been to use an automaton to track the system mode, and then use a different technique to diagnose the continuous behavior (for example, using a set of ARRs for each mode [7], or parameterized ARRs for the complete set of modes [8]). Nevertheless, one of the main difficulties regarding state estimation using these techniques is the need to pre-enumerate the set of *all* possible system-level modes and mode transitions, which is difficult for complex systems. Another approach to fault diagnosis, as shown in [1, 9, 10], qualitatively abstracts the transients in residual deviations and compares them with predicted fault transients. The prediction of fault transients in different modes of the system can also be computationally very expensive.

In order to address the challenges mentioned above, the techniques of *compositional modeling* and *structural model decomposition* have been developed. Building and representing hybrid system models in a *compositional* manner solves the mode pre-enumeration problem. In compositional modeling, the discrete modes are defined at a local level (e.g., at the component level) such that the system-level mode is defined implicitly by the local component-level modes. Since this allows the modeler to focus on the discrete behavior only at the component level, the pre-enumeration of all the system-level modes can be avoided [11, 12]. Additionally, building models in a compositional way facilitates reusability and maintenance, and allows the validation of the components individually before they are composed to create the global hybrid system model.

Structural model decomposition [13] offers another means to decrease the complexity of the hybrid system diagnosis problem [14–16]. In continuous systems diagnosis, structural model decomposition is a popular approach because it allows a global model to be decomposed into local submodels, with each submodel dependent on only a subset of the system faults [13]. As a result, the diagnosis problem becomes much simpler. In hybrid systems, structural model decomposition can significantly decrease the complexity of the problem as well. In addition to minimizing the set of fault effects, each submodel has only a limited number of modes. So instead of reasoning over the exponential number of system-level modes, reasoning need only be performed over the significantly smaller set of submodel modes. Further, structural model decomposition results in *computationally independent* submodels, which lends itself naturally to a *distributed* implementation.

One solution for qualitative fault isolation using structural model decomposition is presented in [17]. Within this approach, however, observation delays are not taken into account and it is applicable only to systems that are modeled using hybrid bond graphs (HBGs). A more efficient model-based methodology for diagnosis, which integrates structural model decomposition within the Hybrid Diagnosis Engine (HyDE), and uses a compositional modeling approach [11], is developed in [18]. The approach demonstrated how the integration of structural model decomposition reduces the computational complexity associated with the fault diagnosis of hybrid systems. The approach presented in this chapter is related to that in [19, 20], but differs in two major ways. First, the former work was based on modeling using HBGs, whereas the modeling framework used here is more general (in which HBGs are a special case). Second, that work was based on a global system model, while in this work, the approach is based on local submodels computed through structural model decomposition.

This chapter presents a model-based, qualitative fault diagnosis framework for hybrid systems, which can diagnose both parametric and discrete faults, and can handle observation delays. The underlying system model is built using a compositional modeling methodology, and structural model decomposition is used to decompose the model into independent submodels, and thus decompose the diagnosis problem and significantly reduce the associated computational complexity. The Advanced Diagnostics and Prognostics Testbed (ADAPT) [21], an electrical power distribution system developed at NASA Ames Research Center, is used as a case study to demonstrate that the approach can correctly isolate faults in hybrid systems even if the system transitions among different mode changes and presents observation delays during the isolation process.

The chapter is organized as follows. Section 8.2 presents the approach to hybrid systems modeling. The diagnosis problem for hybrid systems is formulated in Sect. 8.3 and the qualitative fault isolation approach is presented in Sect. 8.4. Section 8.5 describes the ADAPT case study and presents the experimental results of applying our hybrid fault diagnosis algorithm. Finally, Sect. 8.6 concludes the chapter.

8.2 Hybrid Systems Modeling

Most practical systems demonstrate mixed discrete and continuous dynamics, termed *hybrid dynamics*, and hence, such systems are termed *hybrid systems*. Here, the system can be thought of as inhabiting different operating *modes*, where in each mode there is a specific set of continuous dynamics that governs the system behavior within that mode. The discrete dynamics consist of the behavior governing the transitions between modes.

A circuit example, shown in Fig. 8.1, will be used throughout the chapter to illustrate the approach. The circuit includes a voltage source, V, two capacitors, C_1 and C_2, two inductors, L_1 and L_2, two resistors, R_1 and R_2, and two switches, Sw_1 and Sw_2, connected through a set of series and parallel connections. Sensors measure the current or voltage in different locations (i_3, v_8, and i_{11}, as indicated in Fig. 8.1). Each switch can be in one of two modes: *on* and *off*. Thus, this circuit can be represented as a hybrid system, with four system-level modes.

There are many different modeling formalisms to represent such a system, such as hybrid automata [22] and hybrid bond graphs [23]. From a modeling perspective, it is more convenient to follow a compositional modeling approach, where only local, component-level modes are explicitly defined, and the system-level modes are defined implicitly. In the following, the compositional modeling framework is described, followed by a discussion on causality assignment, and then the structural model decomposition approach.

8.2.1 Compositional Modeling

In a compositional modeling approach, the system is viewed as a set of connected components. Each component is defined by a set of discrete modes, with a different set of constraints describing the continuous dynamics of the component in each mode.

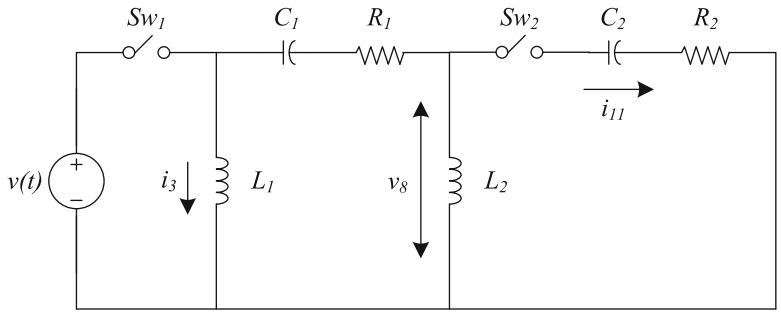

Fig. 8.1 Electrical circuit example

The fundamental building blocks of a hybrid system model are *variables* and *constraints* among those variables. A constraint is defined as follows:

Definition 1 (Constraint) A constraint c is a tuple (ε_c, V_c), where ε_c is an equation involving variables V_c.

A *component* is defined by a set of constraints over a set of variables. The constraints are partitioned into different sets, one for each component mode:

Definition 2 (Component) A *component* κ with n discrete modes is a tuple $\kappa = (V_\kappa, \mathscr{C}_\kappa)$, where V_κ is a set of variables and $\mathscr{C}_\kappa = \{C_\kappa^1, C_\kappa^2, \ldots, C_\kappa^n\}$ is a set of constraints sets, where C_κ^m refers to the set of constraints defining the continuous dynamics in mode m.

Example 1 The components of the circuit are defined in Table 8.1.[1] They include V, C_1, C_2, L_1, L_2, R_1, R_2, Sw_1, Sw_2, as well as components for series and parallel connections.

Example 2 Consider the component Sw_2 (κ_{10}). It has two modes: *off* (represented as mode 1 in Table 8.1) and *on* (represented as mode 2). In the *off* mode, it has three constraints setting each of its currents (i_9, i_{10}, i_{11}) to 0. In the *on* mode, it has also three constraints, setting the three currents equal to each other and establishing that the voltages sum up (it acts like a series connection when in the *on* mode).

A system model is defined as a set of components:

Definition 3 (Model) A *model* $\mathscr{M} = \{\kappa_1, \kappa_2, \ldots, \kappa_k\}$ is a finite set of k components for $k \in \mathbb{N}$.

Example 3 The model of the electrical circuit is made up of all the components detailed in Table 8.1, i.e., $\mathscr{M} = \{\kappa_1, \kappa_2, \ldots, \kappa_{15}\}$. For each component, the variables and constraints are defined for each component mode.

The set of variables for a model $\mathscr{M}$, $V_\mathscr{M}$, is the union of all the component variable sets, i.e., for d components, $V_\mathscr{M} = V_{\kappa_1} \cup V_{\kappa_2} \cup \ldots \cup V_{\kappa_d}$. The interconnection structure of the model is captured using shared variables between components, i.e., components κ_i and κ_j are connected if $V_{\kappa_i} \cap V_{\kappa_j} \neq \varnothing$.

Example 4 In the circuit model, component κ_5 (Series Connection_1) is connected to κ_3 (Parallel Connection_1) through i_4, to κ_6 (R_1) through i_5 and v_5, to κ_7 (C_1) through i_6 and v_6, and κ_8 (Parallel Connection_2) through i_7 and v_7.

The model constraints, $C_\mathscr{M}$, are a union of the component constraints over all modes, i.e., $C_\mathscr{M} = \mathscr{C}_{\kappa_1} \cup \mathscr{C}_{\kappa_2} \cup \ldots \cup \mathscr{C}_{\kappa_d}$. Constraints are exclusive to components, that is, a constraint $c \in C_\mathscr{M}$ belongs to exactly one $\mathscr{C}_\kappa$ for $\kappa \in \mathscr{M}$.

[1] Here, we denote derivatives using dot notation.

Table 8.1 Components of the electrical circuit

Component	Mode	Constraints
κ_1: V	1	$v_1 = u_v$
κ_2: Sw_1	1	$i_1 = 0$
		$i_2 = 0$
	2	$i_1 = i_2$
		$v_1 = v_2$
κ_3: parallel connection_1	1	$v_2 = v_3$
		$v_2 = v_4$
		$i_2 = i_3 + i_4$
κ_4: L_1	1	$\dot{f}_3 = v_3$
		$i_3 = f_3/L_1$
		$f_3 = \int_{t_0}^{t} \dot{f}_3$
κ_5: series connection_1	1	$i_4 = i_5$
		$i_4 = i_6$
		$i_4 = i_7$
		$v_4 = v_5 + v_6 + v_7$
κ_6: R_1	1	$v_5 = i_5 * R_1$
κ_7: C_1	1	$\dot{q}_6 = i_6$
		$v_6 = q_6/C_1$
		$q_6 = \int_{t_0}^{t} \dot{q}_6$
κ_8: parallel connection_2	1	$v_7 = v_8$
		$v_7 = v_9$
		$i_7 = i_8 + i_9$
κ_9: L_2	1	$\dot{f}_8 = v_8$
		$i_8 = f_8/L_2$
		$f_8 = \int_{t_0}^{t} \dot{f}_8$
κ_{10}: Sw_2	1	$i_9 = 0$
		$i_{10} = 0$
		$i_{11} = 0$
	2	$i_9 = i_{10}$
		$i_9 = i_{11}$
		$v_9 = v_{10} + v_{11}$
κ_{11}: R_2	1	$v_{10} = i_{10} * R_2$
κ_{12}: C_2	1	$\dot{q}_{11} = i_{11}$
		$v_{11} = q_{11}/C_2$
		$q_{11} = \int_{t_0}^{t} \dot{q}_{11}$
κ_{13}: $\text{current sensor}_{11}$	1	$i_{11}^* = i_{11}$
κ_{14}: voltage sensor_8	1	$v_8^* = v_8$
κ_{15}: current sensor_3	1	$i_3^* = i_3$

To refer to a particular mode of a model we use the concept of a *mode vector*. A mode vector $\mathbf{m}$ specifies the current mode of each of the components of a model. So, the constraints for a mode $\mathbf{m}$ are denoted as $C_{\mathcal{M}}^{\mathbf{m}}$.

Example 5 Consider a model with five components, then if $\mathbf{m} = [1, 1, 3, 2, 1]$, it indicates that components κ_1, κ_2, and κ_5 use constraints of their mode 1, component κ_3 uses constraints of its mode 3, and component κ_4 uses constraints of its mode 2.

For shorthand, the mode vector will refer to the modes only of the components with multiple modes. So, for the circuit, it refers only to components κ_2 and κ_{10}, resulting in four possible mode vectors, [1 1], [1 2], [2 1], and [2 2].

The switching behavior of each component can be defined using a finite state machine or a similar type of control specification. For the purposes of this chapter, the switching behavior is viewed as a black box where the mode change event is given, and refer the reader to many of the approaches already proposed in the literature for modeling the switching behavior [22, 23]. Although we do not consider state resets explicitly in this paper, this may also be viewed as an output of the switching behavior.

8.2.2 Fault Modeling

A fault is the cause of an unexpected, persistent deviation of the system behavior from the acceptable nominal behavior. In the continuous dynamics, faults are represented as parameter changes in $\Theta_{\mathscr{M}} \subset V_{\mathscr{M}}$, and are termed *parametric faults*.

Definition 4 (Parametric Fault) A parametric fault f is a persistent constant deviation of exactly one parameter $\theta \in \Theta_{\mathscr{M}}$ of the system model $\mathscr{M}$ from its nominal value.

For each parameter, both an increase and a decrease in the parameter value may be considered as a fault. A fault that is an increase in the value of parameter θ is denoted as θ^+, and a fault that is a decrease is denoted as θ^-.

Example 6 In the circuit, $\Theta_{\mathscr{M}} = \{C_1, C_2, L_1, L_2, R_1, R_2\}$. The complete set of parametric faults is $\{C_1^+, C_1^-, C_2^+, C_2^-, L_1^+, L_1^-, L_2^+, L_2^-, R_1^+, R_1^-, R_2^+, R_2^-\}$.

In the discrete dynamics, faults are represented through component mode changes.

Definition 5 (Discrete Fault) A discrete fault f is a persistent change in the mode of exactly one component $\kappa \in \mathscr{M}$ from its nominal value.

Example 7 In the circuit, there are two switching components, Sw_1 and Sw_2, and four associated discrete faults $\{\mathrm{Sw}_1^{off}, \mathrm{Sw}_1^{on}, \mathrm{Sw}_2^{off}, \mathrm{Sw}_2^{on}\}$. Here, the *off* subscript represents the fault where the component changes to the off mode when it should be in the on mode, and the *on* subscript represents the fault where the component changes to the on mode when it should be in the off mode.

8.2.3 Causality

A model is defined without consideration of computational *causality*, i.e., a specification of the computational direction of its constituent constraints. Causality must be considered in order to simulate a model and to study fault propagation.

Given a constraint c, which belongs to a specific mode of a specific component, the notion of a *causal assignment* is used to specify a possible computational direction, or causality, for the constraint c. The causality is indicated by specifying which $v \in V_c$ is the dependent variable in equation ε_c.

Definition 6 (Causal Assignment) A *causal assignment* α_c to a constraint $c = (\varepsilon_c, V_c)$ is a tuple $\alpha_c = (c, v_c^{out})$, where $v_c^{out} \in V_c$ is assigned as the dependent variable in ε_c. We use V_c^{in} to denote the independent variables in the constraint, where $V_c^{in} = V_c - \{v_c^{out}\}$.

In order to assign causality, we must first define which variables within a model are exogenous, i.e., the input variables $U_{\mathscr{M}} \subseteq V_{\mathscr{M}}$. Such variables must always be independent variables in any causal assignment to a constraint involving them, i.e., if a variable v is in $U_{\mathscr{M}}$, then for any constraint c in which $v \in V_c$, it must always be the case that $v \in V_c^{in}$ for any causal assignment.

Parameters that are associated with faults are associated with explicit variables $\Theta_{\mathscr{M}} \in V_{\mathscr{M}}$. They are a special kind of input variable, i.e., $\Theta_{\mathscr{M}} \subseteq U_{\mathscr{M}}$.

In addition, it is useful to refer to a specific set of *output* variables, $Y_{\mathscr{M}} \subseteq V_{\mathscr{M}}$, that are associated with measured outputs of the system.

Example 8 In the circuit, $\Theta_{\mathscr{M}} = \{C_1, C_2, R_1, R_2, L_1, L_2\}$, $U_{\mathscr{M}} = \{u_V\} \cup \Theta_{\mathscr{M}}$, and $Y_{\mathscr{M}} = \{i_3^*, v_8^*, i_{11}^*\}$.

In general, the set of possible causal assignments for a constraint c is as big as V_c, because each variable in V_c can act as v_c^{out}. However, in some cases some causal assignments may not be possible, e.g., if V_c contains any input variables, or if there are noninvertible nonlinear constraints. Also, assuming integral causality, then state variables must always be computed via integration, and so the derivative causality is not allowed. To denote this concept, $\mathbb{A}_c$ refers to the set of permissible causal assignments of a constraint c. For example, for $u \in U_{\mathscr{M}}$, if $u \in V_c$ for some constraint c, (ϵ_c, u) will never be in $\mathbb{A}_c$.

For model $\mathscr{M}$ in mode $\mathbf{m}$, $\mathscr{A}_{\mathscr{M}}^{\mathbf{m}}$ denotes a *complete* set of causal assignments, i.e., for every $c \in C_{\mathscr{M}}^{\mathbf{m}}$, there is exactly one corresponding $\alpha_c \in \mathscr{A}_{\mathscr{M}}^{\mathbf{m}}$. However, only some $\mathscr{A}_{\mathscr{M}}^{\mathbf{m}}$ are actually valid, and this is expressed through the notion of consistency:

Definition 7 (Consistent Causal Assignments) For model $\mathscr{M}$ in mode $\mathbf{m}$, $\mathscr{A}_{\mathscr{M}}^{\mathbf{m}}$ is *consistent* if

(i) for every $c \in C_{\mathscr{M}}^{\mathbf{m}}$, $\alpha_c \in \mathbb{A}_c$, i.e., the causal assignment must be permissible;
(ii) for all $v \in V_{\mathscr{M}} - U_{\mathscr{M}}$, $\mathscr{A}_{\mathscr{M}}^{\mathbf{m}}$ contains exactly one $\alpha = (c, v)$, i.e., every variable that is not an input or parameter is computed by only one (causal) constraint.

Algorithm 1: $\mathscr{A} \leftarrow$ `AssignCausality`$(\mathscr{M}, \mathbf{m}, \mathbb{A})$

```
1:  𝒜 ← ∅
2:  V ← U_ℳ ∪ Θ_ℳ
3:  Q ← U_ℳ ∪ Θ_ℳ ∪ Y_ℳ
4:  for all c ∈ C^m_ℳ do
5:      if |𝔸_c| = 1 then
6:          (c, v) ← 𝔸_c(1)
7:          Q ← Q ∪ v
8:  while |Q| > 0 do
9:      v ← pop(Q)
10:     for all c ∈ C^m_ℳ(v) do
11:         if c ∉ {c : (c, v) ∈ 𝒜} then
12:             α* ← ∅
13:             for all α ∈ 𝔸_c do
14:                 if V_c − {v_α*} ∪ V ≠ ∅ then
15:                     α* ← α
16:                 else if α_v ∈ Y then
17:                     α* ← α
18:                 else if v_α* = v and |C^m_ℳ(v)| − |{c' : (c', v') ∈ 𝒜 ∧ v ∈ v_c}| = 1 then
19:                     α* ← α
20:             if α* ≠ ∅ then
21:                 𝒜 ← 𝒜 ∪ {α*}
22:                 Q ← Q ∪ (V_c − V)
23:                 V ← V ∪ {v_α*}
```

Algorithm 1 describes the causality assignment process for a model given a mode. Causal assignment works by propagating causal restrictions throughout the model. The process starts at inputs, which must always be independent variables in constraints; and outputs, which must always be the dependent variables in at least one constraint. From these variables, we should be able to propagate throughout the model and compute a valid causal assignment for the model in the given mode. For the purposes of this paper, we assume integral causality and that the model possesses no algebraic loops.[2] In this case, there is only one valid causal assignment.

Specifically, the algorithm works as follows. It keeps queue of variables to propagate causality restrictions, Q, and a set of variables that are computed in the current causality, V. Initially, V is set to U, because these variables are not to be computed by any constraint. Q is set to U and Y, since the causality of constraints is restricted to U variables being independent variables and Y variables being dependent variables. We add also to Q any variables involved in constraints that have only one permissible causal assignment, because this will also restrict other causal assignments. The set of causal assignments is maintained in $\mathscr{A}$.

The algorithm goes through the queue, inspecting variables. For a given variable, we obtain all constraints it is involved in, and for each one that does not yet have a causal assignment (in $\mathscr{A}$), we go through all permissible causal assignments, and

[2] If algebraic loops exist, the algorithm will terminate before all constraints have been assigned a causality. Extending the algorithm to handle algebraic loops is similar to that for bond graphs; a constraint without a causality assignment is assigned one arbitrarily, and then effects of this assignment are propagated until nothing more is forced. This process repeats until all constraints have been assigned causality.

determine if the causality is forced into one particular causal assignment, α^*. If so, we assign that causality and propagate by adding the involved variables to the queue. A causal assignment $\alpha = (c, v)$ is forced in one of three cases: (i) v is in Y, (ii) all variables other than v of the constraint are already in V, and (iii) v is not yet in V, and all but one of the constraints involving v have an assigned causality, in which case no constraint is computing v and there is only one remaining constraint that must compute v. The algorithm must visit all constraints and does not backtrack, so the time complexity is linear in the size of the model.

Example 9 Consider the mode $\mathbf{m} = [1\ 2]$. Here, $\mathscr{A}^{[1\ 2]}$ is given in column 4 of Table 8.1, denoted by the v_c^{out} in the causal assignment. In this mode, the first switch is off, so i_1 and i_2 act as inputs. Given the integral causality assumption, a unique causal assignment to the model exists and is specified in the column.

Example 10 Consider the mode $\mathbf{m} = [2\ 1]$. Here, $\mathscr{A}^{[2\ 1]}$ is given in column 8 of Table 8.1. In this mode, the second switch is off, so i_9, i_{10}, and i_{11} act as inputs. Given the integral causality assumption, a unique causal assignment to the model exists and is specified in the column. Note that some causal assignments are in the same as in $\mathbf{m} = [1\ 2]$, while others are different.

When the system mode changes, causality can be recomputed using Algorithm 1. More efficient, incremental, causality assignment, based on the assignment of the previous mode, can also be performed [14], however that is beyond the scope of this chapter.

8.2.4 Structural Model Decomposition

Structural model decomposition creates local submodels given a system model. For a hybrid system, the mode of the system must be specified. When the mode changes, the derived submodels may also change, i.e., if they include constraints of components that have changed mode. This section describes how submodels are generated. Their application to diagnosis will be described in Sect. 8.4.

The procedure for generating a submodel from a causal model is given as Algorithm 2 [13]. The following applies to a given mode, so in the remainder of the section the mode superscript is dropped. Given a causal model $\mathscr{M}$, and an output variable to be computed y, the `GenerateSubmodel` algorithm derives a causal submodel $\mathscr{M}_i$ that computes y using as local inputs only variables from $U^* = U \cup (Y - \{y\})$. We briefly summarize the algorithm below.

In Algorithm 2, the *variables* queue represents the set of variables that have been added to the submodel but have not yet been resolved, i.e., they cannot yet be computed by the submodel. This queue is initialized to $\{y\}$, and the algorithm then iterates until this queue has been emptied, i.e., the submodel can compute y using only variables in U^*. For each variable v that must be resolved, we use Subroutine 3

Algorithm 2: $\mathscr{M}_i = \texttt{GenerateSubmodel}(\mathscr{M}, U^*, V^*)$

```
1:  V_i ← V*
2:  C_i ← ∅
3:  A_i ← ∅
4:  variables ← V*
5:  while variables ≠ ∅ do
6:      v ← pop(variables)
7:      c ← GetBestConstraint(v, V_i, U*, A)
8:      C_i ← C_i ∪ {c}
9:      A_i ← A_i ∪ {(c, v)}
10:     for all v' ∈ V_c do
11:         if v' ∉ V_i and v' ∉ Θ and v' ∉ U* then
12:             variables ← variables ∪ {v'}
13:         V_i ← V_i ∪ {v'}
14: M_i ← (V_i, C_i, A_i)
```

Algorithm 3: $c = \texttt{GetBestConstraint}(v, V_i, U^*, \mathscr{A})$

```
1:  C ← ∅
2:  c_v ← find c where (c, v) ∈ A
3:  if V_{c_v} ⊆ V_i ∪ U* then
4:      C ← C ∪ {c_v}
5:  for all y ∈ Y ∩ U* do
6:      c_y ← find c where (c, y) ∈ A
7:      if v ∈ V_{c_y} and V_{c_y} ⊆ V_i ∪ U* ∪ Θ then
8:          C ← C ∪ {c_y}
9:  for all y ∈ Y ∩ U* do
10:     c_y ← find c where (c, y) ∈ A
11:     V' ← V_{c_y} − {y}
12:     for all v' ∈ V' do
13:         c_{v'} ← find c where (c, v') ∈ A
14:         if v ∈ V_{c_{v'}} and V_{c_y} ⊆ {v} ∪ U* ∪ Θ then
15:             C ← C ∪ {c_{v'}}
16: if C = ∅ then
17:     c ← c_v
18: else if c_v ∈ C then
19:     c ← c_v
20: else
21:     C' ← C
22:     for all c_1, c_2 ∈ C where c_1 ≠ c_2 do
23:         y_1 ← find y where (c_1, y_1) ∈ A
24:         y_2 ← find y where (c_2, y_2) ∈ A
25:         if (y_1 ◁ y_2) ∈ P then
26:             C' ← C' − {c_1}
27:     c ← first(C')
```

(`GetBestConstraint` subroutine) to find the constraint that should be used to resolve v in the minimal way.

The `GetBestConstraint` subroutine, given as Algorithm 3, (which has been updated from [13]) tries to find a constraint that *completely* resolves the variable, i.e., resolves v without further backward propagation (all other variables involved in the constraint are in $V_i \cup \Theta \cup U^*$). Such a constraint may be the one that computes v in the current causality, if all needed variables are already in the submodel (in V_i) or are available local inputs (in U^*); such a constraint may be one that computes a

measured output $y^* \in U^*$, in which case the causality will be modified such that y^* becomes an input, i.e., the constraint in the new causality will compute v rather than y^*; or such a constraint may be one that computes some y^* through some v' in an algebraic relation. If no such constraint exists, then the constraint that computes v in the current causal assignment is chosen, and further backward propagation will be necessary. A preferences list, P, is used to break ties if multiple minimal constraints exist to resolve v.

For a given causal model in a given mode, we have the equivalent of a continuous systems model for the purpose of structural model decomposition, and we can compute minimal submodels using the `GenerateSubmodel` algorithm described in previous work [13]. The algorithm finds a submodel, which computes a set of local outputs given a set of local inputs, by searching over the causal model. It starts at the local inputs, and propagates backwards through the causal constraints, finding which constraints and variables must be included in the submodel. When possible, causal constraints are inverted in order to take advantage of local inputs. In the worst case, the algorithm ends up with the global model, so would traverse the entire causal structure. Additional information and the pseudocode are provided in [13].

The local inputs for a submodel are selected from variables for which the values will be known. This includes variables in $U_{\mathscr{M}}$, which are assumed to be known, but could also include variables in $Y_{\mathscr{M}}$, because the values of these are being provided by sensors. On average, `GenerateSubmodel` will find a submodel that is a small subset of the global model. In the worst case, if no decomposition is possible, it will return the global model, minus the other outputs. However, in this case, this submodel is still computationally independent of the others and can still be run in parallel.

Example 11 Submodels can be represented visually using a graph notation, where vertices correspond to variables, and edges correspond to constraints with causal assignments, i.e., a directed edge from v_i to v_j means that v_j is computed using v_i. Consider a submodel for which the local output is i_3^*, and the available local inputs are $\{v_8^*, i_{11}^*\} \cup U_{\mathscr{M}}$. The submodel graphs for two modes are shown in Fig. 8.2. If Sw_1 is on with Sw_2 off, for example, i_3^* can be determined completely by u_V (see Fig. 8.2b). If Sw_1 is off with Sw_2 on, then it must be computed based on the value of v_8^* (see Fig. 8.2a).

The main advantage of structural model decomposition is that faults appear in only a subset of the submodels. Following a model-based approach, in which the submodels are used to compute the nominal system behavior, only a subset of the submodels will be affected by a single fault. Thus, the reasoning becomes much simpler.

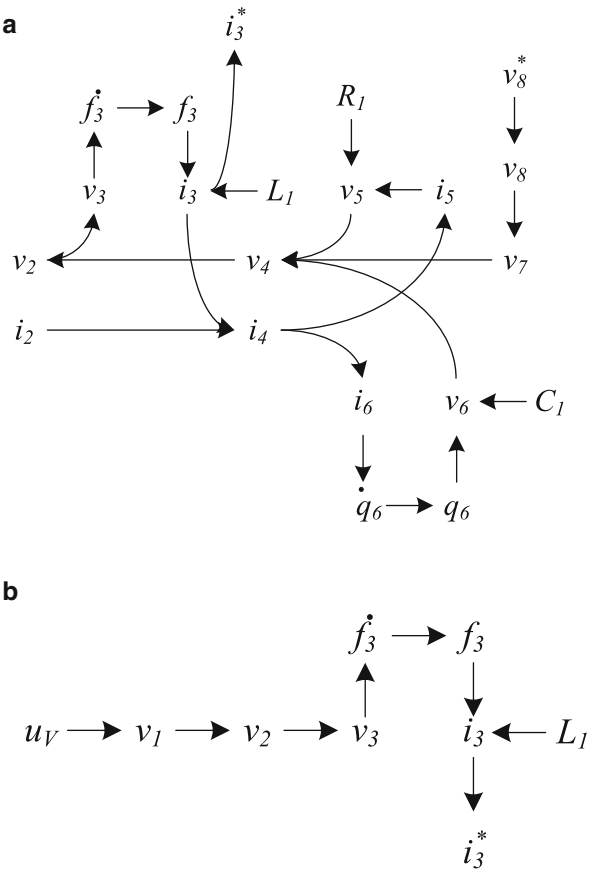

Fig. 8.2 Submodel graphs for i_3^*. (**a**) i_3^* submodel graph in the mode with the first switch off and the second on. (**b**) i_3^* submodel graph in the mode with the first switch on and the second off

8.3 Problem Formulation

The diagnosis problem is one of mapping observations on a system to an explanation for that set of observations, specifically, which faults may have occurred to produce those observations. In general terms, a fault is a single change in the system. Here, we make the single-fault assumption.

Assumption 1 Only single faults occur in the system.

That is, we assume that only a single change in the system has occurred and can explain the given observations. Since faults are typically unlikely to occur in the first place, when faults are independent of each other, the probability of multiple faults is extremely low. So, the single-fault assumption is common in practical settings.

As described in Sect. 8.2, faults may be either parametric (represented as increases or decreases in system parameter values) or discrete (represented as changes in the modes of components). Further, we assume that faults are persistent, i.e., once a fault occurs, it remains.

Assumption 2 Faults are persistent.

Generally speaking, for the purposes of diagnosis, we consider an observation to be an event observed at a particular time.

Definition 8 (Observation) An *observation* is a tuple (e, t), where e is an observed event and t is the time of observation.

Two kinds of events are considered, mode changes and fault signatures. We define mode change events specific to components.

Definition 9 (Mode Change Event) An event (κ, m) represents component κ changing to its mode m.

For the purposes of this chapter, we assume these are known/observable, i.e., they are considered an input to our system.

Assumption 3 (Mode Change Observability) All mode change events are observable.

Following a qualitative fault isolation approach, the remaining events take the form of qualitative symbols representing the transients caused by faults, termed fault signatures.[3] These symbols are computed from system residuals, i.e., the differences between observed and model-predicted outputs.

Definition 10 (Residual) A residual r for output y as measured by a sensor is computed as $r = y - y^*$, where y^* is the model-predicted output value.

Under the single-fault assumption, a *diagnosis* is simply a fault that is consistent with a given observation sequence.

Definition 11 (Diagnosis) For a system with fault set F, and a sequence of observations O, a *diagnosis* for O, d_O is a fault $f \in F$ that is consistent with O. The set of all diagnoses for O is denoted as D_O.

The diagnosis problem can then be formally defined as follows.

Problem 1 For a system with fault set F, given a finite sequence of observations O, find the set of diagnoses $D_O \subseteq F$.

[3] Fault signatures will be defined formally in Sect. 8.4.

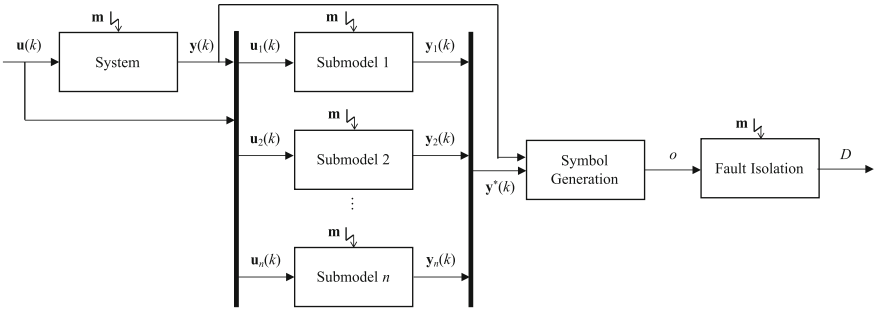

Fig. 8.3 Diagnosis architecture

8.3.1 Architecture

To solve this problem, the architecture shown in Fig. 8.3 is used. The inputs, $\mathbf{u}(k)$, are fed into the system, and the system produces measured outputs, $\mathbf{y}(k)$. These signals are then decomposed into local input and outputs for the local submodels. For each sensor, one submodel is defined where the corresponding output variable is the single local output of the submodel, and all other output variables may serve as local inputs along with $U_{\mathscr{M}}$. Thus, for a set of n sensors, there will be n submodels. The mode change $\mathbf{m}$ is also fed into the system and the submodels.

If a mode change occurs, the system changes mode and the submodels also change modes to reflect the new system mode. This requires regenerating the submodels due to the change in causality, and this can be done efficiently using the causality reassignment algorithm.

The actual system outputs, $\mathbf{y}(k)$, along with the submodel-generated outputs, $\mathbf{y}^*(k)$ are then fed to the symbol generation module. Following a qualitative fault isolation approach, the residuals are transformed into qualitative `0` (no change), $-$ (decrease), and + (increase) changes for the magnitude and slope in the residual. Once a residual is detected to deviate in a statistically significant manner from zero, symbols are generated for that residual, and fed into the fault isolation module.

The fault isolation module reasons over the sequence of observations, consisting of these qualitative symbols and mode change events, to isolate the fault. The algorithms underlying the fault isolation module are described in the following section.

8.4 Qualitative Fault Isolation for Hybrid Systems

As described in Sect. 8.3, the diagnosis problem is to map a sequence of fault signatures and mode change events to single faults that are consistent with the sequence. At the core of the qualitative fault isolation approach is the concept of a

fault signature. In this section, we first describe fault signatures. The fault isolation procedure is then described, followed by a discussion of scalability.

8.4.1 Fault Signatures

The basis of the qualitative fault isolation approach is the concept of a fault signature [10].

Definition 12 (Fault Signature) A *fault signature* for a fault f and residual r in mode m, denoted by $\sigma_{f,r,m}$ is a set of symbols representing changes in r caused by f at the point of the occurrence of f in mode m. The set of all fault signatures for a fault f over residuals R in mode m is denoted as $\Sigma_{f,R,m}$.

In this work, fault signatures are made up of a set of two symbols: the qualitative change in residual magnitude, and the qualitative change in residual slope. Each one of these symbols can take the values + (increase), − (decrease), and 0 (no change). These symbols are based on the transient that is produced when a fault occurs [9]. We write always the magnitude symbol followed by the slope symbol, e.g., a signature +− represents an increase in magnitude and a decrease in slope.

A fault signature provides a prediction of the observation that will be made for a system in a particular mode when that fault happens. For the case of a parametric fault, this is a straightforward concept and we refer the reader to previous works [24]. For discrete faults, the interpretation of the fault signature remains the same, although a discrete fault will change the mode. Specifically, if the system is in mode m and a discrete fault f occurs (thus changing the mode), the signatures in $\Sigma_{f,R,m}$ will be those observations predicted for the fault occurring in mode m, and not the mode in which the fault drives the system into. So, if we know the system is in mode m and fault signatures are observed, we always look in $\Sigma_{f,R,m}$ for every $f \in F$ to reason about which fault has occurred.

Example 12 Table 8.2 shows the fault signatures for the circuit example for the two modes considered for the local submodel residuals. Consider that fault L_1^- occurs in the system. In $\mathbf{m} = [2\ 1]$, it affects only the residual for i_3^*, as it is the only local submodel where it appears (see Table 8.2). In $\mathbf{m} = [2\ 1]$, it also affects i_3^*. The fault R_1^+ will also, but they can be distinguished by the specific change produced by the fault. For L_1^-, a +− is produced, whereas for R_1^+, a 0− is produced.

Fault signatures can be derived from analysis of the system model [9, 19] or via simulation. Here, we assume they are given as input.

Since we have a single submodel for each residual, fault isolation within a single mode is straightforward. Given an observed sequence of fault signatures, Σ in mode m, we determine which faults match all signatures in Σ.

Example 13 For the circuit, given $\mathbf{m} = [1\ 2]$ and $\Sigma = \{r_{v_8^*}^{+-}\}$, then $D = \{C_1^-, L_2^-\}$. Note that the $*$ symbol may match either + or −.

For the purposes of this paper, we assume that signatures are correctly observed.[4]

Assumption 4 (Correct Observation) If a fault f occurs in mode m, then if the system does not change mode after the occurrence of the fault, the observed fault signatures will belong to $\Sigma_{f,R,m}$.

8.4.2 Hybrid Systems Diagnosis

For hybrid systems, fault signatures are always given as a function of the system mode. If there is no mode change occurring between the point of fault occurrence and the diagnosis of the fault, then the problem reduces to the continuous systems case. Otherwise, some combination of fault signatures from different modes may be observed, depending on when the mode changes take place and how long it takes for fault signatures to manifest.

Example 14 Consider the residuals in Table 8.2. Assume that the system starts in $\mathbf{m} = [1\ 2]$ and R_1^+ occurs. Then we could observe $r_{i_3^*}^{0-}$. Now, assume that the system moves to mode $\mathbf{m} = [2\ 1]$, now we would observe $r_{v_8^*}^{+-}$. This set of fault signatures is not found in any single mode, so the reasoning must extend over the sequence of mode changes.

As shown in the example, the first challenge of the approach for hybrid systems is that now the observed fault signatures may correspond to different modes. Thus, the fault isolation process must span over several potential mode changes. By knowing the mode of the system, we can know which set of fault signatures corresponds to the predicted observations for each fault.

The advantage of structural model decomposition here is that such combinations are limited and easier to deal with compared to an approach using a single global model, where potentially all residuals may be affected by every fault. In that case,

Table 8.2 Fault signatures for minimal submodels of the electrical system

Mode	$\mathbf{m} = [1\ 2]$			$\mathbf{m} = [2\ 1]$		
Fault	$r_{i_{11}^*}$	$r_{i_3^*}$	$r_{v_8^*}$	$r_{i_{11}^*}$	$r_{i_3^*}$	$r_{v_8^*}$
C_1^-	00	0+	00	00	00	−+
C_2^-	00	00	−0	00	00	00
L_1^-	00	+−	00	00	+0	00
L_2^-	−0	00	00	00	00	−∗
R_1^+	00	0−	00	00	00	+−
R_2^+	00	00	+0	00	00	00

[4] Relaxation of this assumption has been explored for continuous systems in [25].

there are many potential combinations and increased ambiguity. The decoupling of faults and residuals provided by structural model decomposition helps to reduce this complexity.

As discussed in the previous section, in this work we assume that all commanded mode change event are observable. However, even if we know the current mode of the system, there is another related layer of complexity to consider, the *observation delay*, which refers to the delayed observation of fault signatures. The difficulty relies in that the system may be in one mode, but when the observation arrives it might have moved to a different system mode, thus the mode in which the observation was made may not be known exactly.

The observation delay can be manifested in different ways. For example, fault detection in our framework is done by checking if the residual crosses a threshold. Due to the presence of noise and in order to perform a robust fault detection, we use a statistical test that computes the mean of the residual over a small time window and check if that mean has crossed the threshold. In practice, this means that the signal could actually cross the threshold in one mode, but the mean of the signal could cross only in the next mode. Thus, the observation of this signature is delayed. In this work we assume that the observation delay is finite and bounded.

Assumption 5 (Bounded Observation Delay) The delay of any observation is no greater than Δ.

Given our assumptions, the algorithm for a single step of fault isolation for hybrid systems is shown as Algorithm 4. Note that we reason through fault signatures the same for discrete and for parametric faults, hence the algorithm presented is the same for both situations. As inputs, the algorithm takes the current diagnosis, D_i, the previous sequence of fault signatures, λ_i, the new fault signature, σ_{i+1}, and the set of recent modes that falls within $[t-\Delta, t]$, $M_{r,\Delta}$, for the submodel that generates residual r. The main difference of this algorithm against the previous version for continuous systems is that we need to check signatures for each one of the recent modes.

Another advantage here of structural model decomposition is that the set of recent modes is dependent on the model used for fault isolation, and consequently is a function of the residual associated with the signature. If a global model is used, the residual generator will contain all system modes. However, if local submodels are used, the residual generator will only contain the local modes of that submodel (which is always less than the number of system modes). Thus, fewer modes must be searched and efficiency is improved.

Algorithm 4: $D_{i+1} = \texttt{FaultIsolation}(D_i, \lambda_i, \sigma_{i+1}, M_{r,\Delta})$

```
1: D_{i+1} ← ∅
2: for all q ∈ M_{r,Δ} do
3:     for all f ∈ D_i ∩ F_{r,q} do
4:         if σ_{i+1} ∈ Σ_{f,r,σ_{i+1},m} then
5:             D_{i+1} ← {f}
```

If the signature is consistent in any of the modes, it must be added to D_{i+1}. Here, for a given mode m, we need to check only the subset of faults that are included in the current diagnosis and can actually affect this residual in this mode, denoted as $F_{r,m}$. An observed signature, σ_{i+1}, is consistent with a fault if the predicted signature for its residual ($r_{\sigma_{i+1}}$) is included in the signature set for that fault and residual in the given mode.

Algorithm 4 just presents a single reasoning step, when there is a new observed signature, of the fault isolation process. When implemented, this algorithm would be placed within a general progressive monitoring algorithm that keeps track of the current diagnosis, and computes the set of recent modes based on the times events are observed. In the worst case, it must check consistency with all faults and all deviated residuals for all given mode changes, so in the worst case is $O(|F||R||M_r|)$. On average, it is much less, since the candidate set reduces with each newly observed fault signature.

8.4.3 Scalability

The complexity of the fault isolation algorithm is dependent on the number of faults, $|F|$, the number of residuals, $|R|$, and the number of modes, $|M|$. For the global model case, all faults, residuals, and modes in $M_{r,\Delta}$ must be searched. Because r is computed using the global model, it is a function of the system-level mode. For an n-tank system, there are $n-1$ switching components and so 2^{n-1} system-level modes. Clearly, diagnosis in this case will not scale.

For the local submodel case, each residual is generated by a minimal submodel, so it improves over the global model approach by simultaneously reducing both the effective $|R|$ and the effective $|M|$. The effective $|R|$ is decreased because with structural model decomposition each fault affects only a subset of the residuals, so for each new residual deviation only a subset of faults needs to be checked for consistency. The effective $|M|$ is reduced because with structural model decomposition each residual is reconfigured only based on a few local component modes, whereas for the global model each residual is dependent on the system-level modes (which increases exponentially with the number of switching components). Due to these properties of structural model decomposition, the complexity grows at a significantly smaller rate as the system size increases than with the global model approach.

8.5 Case Study

The Advanced Diagnostics and Prognostics Testbed (ADAPT) is an electrical power distribution system that was built to mimic the operation of such systems on spacecraft [21]. Through the International Diagnostic Competition (DXC), it has

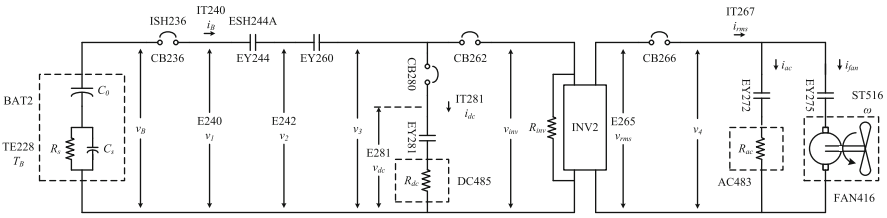

Fig. 8.4 ADAPT-Lite schematic

Table 8.3 Components of ADAPT-Lite and their failure modes

Components	Failure mode
AC483, DC485	Resistance offset
E240, E242, E265, E281, TE228, IT240, IT267, IT281, ST516	Offset
ESH244A, ISH236	Stuck
EY244, EY260, EY272, EY275, EY284	Stuck open
	Stuck closed
FAN416	Underspeed
	Overspeed

been established as a diagnostic benchmark system [26–28]. The diagnosis approach is applied to a subset of ADAPT, called ADAPT-Lite.

Figure 8.4 provides a system schematic for ADAPT-Lite. A battery (BAT2) supplies electrical power to several loads, passing through several circuit breakers (CB236, CB262, CB266, and CB280), and controlled by relays (EY244, EY260, EY281, EY272, and EY275). An inverter (INV2) converts dc to ac power. ADAPT-Lite has one dc load (DC485) and two ac loads (AC483 and FAN416). Sensors report electrical voltage (names beginning with "E"), electrical current ("IT"), and the positions of relays and circuit breakers ("ESH" and "ISH", respectively). There is one sensor to report the operating state of a load (fan speed, ST516) and another to report the battery temperature (TE228).

Table 8.3 summarizes component fault modes in ADAPT-Lite. The resistance faults, fan speed faults, and sensor faults are modeled as parametric faults, and the remaining faults are modeled as discrete faults.

8.5.1 System Modeling

Following the component-based modeling approach outlined in Sect. 8.2, each component of the system is represented as a set of modes and constraints for each mode. The components are one of the following types: battery, relay, inverter, dc load, ac load, fan, and sensor. Since no faults are considered for the circuit breakers,

they are omitted from the model. In the following, the component models are each described in turn.

BAT2 consists of two 12 V lead-acid batteries in series, which are lumped together into a single battery model. A simplified electrical circuit equivalent model, consisting of a single large capacitance, C_0, in series with a capacitor-resistor pair, C_s and R_s, that subtracts from the voltage provided by C_0 (see Fig. 8.4), is used. The battery may then be described as

$$\dot{v}_0 = \frac{1}{C_0}\left(-i_B\right), \tag{8.1}$$

$$v_0 = \int_{t_0}^{t} \dot{v}_0 dt, \tag{8.2}$$

$$\dot{v}_s = \frac{1}{C_s}\left(i_B R_s - v_s\right), \tag{8.3}$$

$$v_s = \int_{t_0}^{t} \dot{v}_s dt, \tag{8.4}$$

$$v_B = v_0 - v_s, \tag{8.5}$$

where i_B is the battery current, v_B is the battery voltage, v_0 is the voltage across C_0, and v_s is the voltage drop across C_s and R_s. The battery temperature is assumed constant, i.e.,

$$\dot{T}_B = 0, \tag{8.6}$$

$$T_B = \int_{t_0}^{t} \dot{T}_B dt. \tag{8.7}$$

The relays each have two modes, on and off. When off, the constraints are:

$$i_l = 0 \tag{8.8}$$

$$v_r = 0 \tag{8.9}$$

$$p = 0, \tag{8.10}$$

where i_l is the current on the left side of the relay, v_r is the voltage on the right side, and p is the position. When on, constraints are:

$$i_l = i_r \tag{8.11}$$

$$v_l = v_r \tag{8.12}$$

$$p = 1, \tag{8.13}$$

where i_r is the current on the right side, and v_l is the voltage on the left side. When on, the voltages and currents on either side of the relay must be equal. When off, the current on the left side is set to zero, while the current on the right is determined by the component on the right. Similarly, the voltage on the right is set to zero, while the voltage on the left is determined by the component on the left. Following this modeling convention, voltage is always determined by the component on the left, and current by the component on the right, no matter which mode the system is in, and with consistent causality assignment in each mode.

The dc load is a simple resistance:

$$v_{\text{dc}} = i_{\text{dc}} \cdot R_{\text{dc}}, \tag{8.14}$$

where v_{dc} is the voltage across the load, i_{dc} is the current through the load, and R_{dc} is the load resistance. Similarly, the ac load is also a simple resistance:

$$v_{\text{ac}} = i_{\text{ac}} \cdot R_{\text{ac}}, \tag{8.15}$$

however the corresponding voltage and current are rms values.

The fan current is a function of the applied voltage:

$$v_{\text{fan}} = i_{\text{fan}} \cdot R_{\text{fan}}, \tag{8.16}$$

where R_{fan} is the magnitude of the fan impedance, and v_{fan} and i_{fan} are the rms voltage and current, respectively. The fan speed is expressed as a function of its current

$$\dot{\omega} = \frac{1}{J_{\text{fan}}} \left(i_{\text{fan}} \cdot g_{\text{fan}} - \omega \right), \tag{8.17}$$

$$\omega = \int_{t_0}^{t} \dot{\omega} \, dt, \tag{8.18}$$

where J_{fan} is an inertia parameter and g_{fan} is a gain parameter.

The inverter transforms dc power to ac power. When operating nominally, the rms voltage v_{rms} is controlled very close to 120 V ac as long as the input voltage is above 18 V:

$$v_{\text{rms}} = 120 \cdot (v_{\text{inv}} > 18). \tag{8.19}$$

From a power balance of the ac and dc sides of the inverter, it results that $v_{\text{inv}} \cdot i_{\text{inv}} = e \cdot v_{\text{rms}} \cdot i_{\text{rms}}$, where e is the inverter efficiency, i_{rms} is the inverter rms current, v_{inv} is the inverter voltage on the dc side, and i_{inv} is the input dc current to the inverter. The inverter still draws a small amount of current even when $i_{\text{rms}} = 0$, and this is captured as a dc resistance parallel to the inverter, R_{inv}. Hence, the following equation is derived:

$$i_{\text{inv}} = \frac{v_{\text{rms}} \cdot i_{\text{rms}}}{e \cdot v_{\text{inv}}} + \frac{v_{\text{inv}}}{R_{\text{inv}}}. \tag{8.20}$$

The sensors are modeled using a bias term. For sensor s, the constraint is:

$$y_s = s_s + b_s, \tag{8.21}$$

where s_s is the raw signal value, b_s is the bias term, and y_s is the sensor output.

Component models are instantiated and connected according to Fig. 8.4 through series and parallel connections. With five relays, each with two modes, there are a total of $2^5 = 32$ system modes.

The three loads (dc, ac, and fan) each have a single parametric fault, represented through their respective resistance parameters. Further, each of the eleven sensors has an offset fault represented as a change in the bias parameter. Discrete faults are also associated with each of the five relays. A relay can turn on/off without a command, or fail to turn on/off in response to a command. So for a single system mode, there are 19 potential faults that may occur.

8.5.2 *Structural Model Decomposition*

As described in Sect. 8.3, one submodel is defined for each sensor. In general, there may be a different submodel for each system-level mode, however, many of these submodels are the same for different system-level modes, because the behavior of many of the switching components isolated from a given submodel due to the decomposition.

Example 15 Consider the submodel for E281. It has only two modes: one in which the voltage is determined by the measured value of IT281 (and EY281 is on), and one in which is set to zero (and EY281 is off). Its mode depends only on the state of relay EY281. When off, the submodel consists of the following constraints:

$$\begin{aligned}
v_{r,\text{EY260}} &= 0 \\
v_{l,\text{P1}} &= v_{r,\text{EY260}} \\
v_{r,2,\text{P1}} &= v_{l,\text{P1}} \\
v_{l,\text{CB280}} &= v_{r,2,\text{P1}} \\
v_{r,\text{CB280}} &= v_{l,\text{CB280}} \\
v_{l,\text{EY284}} &= v_{r,\text{CB280}} \\
s_{\text{E281}} &= v_{l,\text{EY284}} \\
y_{\text{E281}} &= b_{\text{E281}} + s_{\text{E281}},
\end{aligned}$$

where P1 refers to a parallel connection with the 1 and 2 subscripts indicating the connection. The only local input is b_{E281}, which is assumed to be 0 in the nominal case. When on, the submodel consists of the following constraints:

$$\begin{aligned} s_{\text{E242}} &= -b_{\text{E242}} + y_{\text{E242}} \\ v_{r,\text{EY260}} &= s_{\text{E242}} \\ v_{l,\text{P1}} &= v_{r,\text{EY260}} \\ v_{r,2,\text{P1}} &= v_{l,\text{P1}} \\ v_{l,\text{CB280}} &= v_{r,2,\text{P1}} \\ v_{r,\text{CB280}} &= v_{l,\text{CB280}} \\ v_{l,\text{EY284}} &= v_{r,\text{CB280}} \\ s_{\text{E281}} &= v_{l,\text{EY284}} \\ y_{\text{E281}} &= b_{\text{E281}} + s_{\text{E281}}. \end{aligned}$$

The local inputs are b_{E281}, which is assumed to be zero, b_{E242}, which is assumed to be zero, and y_{E242}, which is the measured value of E242.

Over the whole system, each submode has between 1 and 4 modes. This greatly simplifies the required diagnostic reasoning, since any given mode change will only affect a minimal number of submodels.

8.5.3 Diagnosability

Given any mode of the system, fault signatures can be derived from any fault that may occur within that mode through the qualitative fault propagation algorithms. Signatures for the mode in which all relays are on are shown in Table 8.4. For space, only the signature for parametric faults increasing in the positive direction is shown (for the negative direction, the signature signs are flipped). For example, a bias fault in E265 will produce a deviation in the residual for E265, along with those for IT240, IT267, and ST516, because the value of E265 is used as a local input in the submodels for those sensors. A fault in the resistance of AC483, in contrast, will produce a change only in the residual of IT267, since it appears only in the submodel for IT267. Note that a $*$ symbol is used to denote an indeterminate effect (i.e., the sign of the change depends on the system state).

It is important to note here that some ambiguity in the fault isolation results is expected, i.e., the system is not fully diagnosable. For example, a resistance offset in DC485 will result in a single change in the residual of IT281. A bias in that sensor could result in the same signature, along with a bias in E281 and EY281 turning off. So if the resistance fault occurs, the change in IT281 will be observed, resulting in all

Table 8.4 Fault signatures for faults occurring in the mode with all relays on

Fault	r_{E240}	r_{E242}	r_{E265}	r_{E281}	r_{ESH244A}	r_{ISH236}	r_{IT240}	r_{IT267}	r_{IT281}	r_{ST516}	r_{TE228}
$EY244^{off}$	0 0	−∗	0 0	0 0	−0	0 0	−∗	0 0	0 0	0 0	0 0
$EY260^{off}$	0 0	−∗	−∗	−∗	0 0	0 0	−∗	0 0	0 0	0 0	0 0
$EY272^{off}$	0 0	0 0	0 0	0 0	0 0	0 0	0 0	−∗	0 0	0 0	0 0
$EY275^{off}$	0 0	0 0	0 0	0 0	0 0	0 0	0 0	−∗	0 0	0−	0 0
$EY281^{off}$	0 0	0 0	0 0	0 0	0 0	0 0	−∗	0 0	−∗	0 0	0 0
R^{+}_{AC483}	0 0	0 0	0 0	0 0	0 0	0 0	0 0	−0	0 0	0 0	0 0
R^{+}_{DC485}	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	−0	0 0	0 0
R^{+}_{RFAN416}	0 0	0 0	0 0	0 0	0 0	0 0	0 0	−0	0 0	0−	0 0
b^{+}_{E240}	+0	−0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
b^{+}_{E242}	0 0	+0	−0	−0	0 0	0 0	∗0	0 0	0 0	0 0	0 0
b^{+}_{E265}	0 0	0 0	+0	0 0	0 0	0 0	∗0	∗0	0 0	0∗	0 0
b^{+}_{E281}	0 0	0 0	0 0	+0	0 0	0 0	0 0	0 0	∗0	0 0	0 0
b^{+}_{ESH244A}	0 0	0 0	0 0	0 0	+0	0 0	0 0	0 0	0 0	0 0	0 0
b^{+}_{ISH236}	0 0	0 0	0 0	0 0	0 0	+0	0 0	0 0	0 0	0 0	0 0
b^{+}_{IT240}	0 0	0 0	0 0	0 0	0 0	0 0	+0	0 0	0 0	0 0	0 0
b^{+}_{IT267}	0 0	0 0	0 0	0 0	0 0	0 0	∗0	+0	0 0	0 0	0 0
b^{+}_{IT281}	0 0	0 0	0 0	0 0	0 0	0 0	−0	0 0	+0	0 0	0 0
b^{+}_{ST516}	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	+0	0 0
b^{+}_{TE228}	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	+0

those faults as diagnoses. No further residuals will deviate to further reduce the fault set. If time limits are set for how long to wait to observe further deviations, then this would improve the diagnosability and allow this fault to be uniquely isolated [24].

In general, the diagnosability results from using residuals from local submodels generated from structural model decomposition may not be the same as using those from a global model, as is proven in [29]. In practice, the diagnosability should be compared to determine if there is any loss of diagnosability from using structural model decomposition.

8.5.4 Results

As an example to illustrate the diagnosis process, consider the initial mode 11100 (here, the mode is designated by the sequence of relay states, for the relays in alphabetical ordering), i.e., EY244, EY260, and EY275 are on, so power is sent only to the ac load. The fault, EY244 turning off uncommanded, occurs at 120.0 s. At 121.0 s, a decrease in the residuals for both y_{E242} and $y_{ESH244A}$ is detected (see Fig. 8.5). Considering the decrease first in y_{E242}, the initial diagnoses are $\{b^{+}_{E240}, EY260^{off}, EY244^{off}, b^{-}_{E242}\}$. Considering next the decrease in $y_{ESH244A}$, we can reduce the diagnoses to $EY244^{off}$, which is the true fault; only a fault in EY244

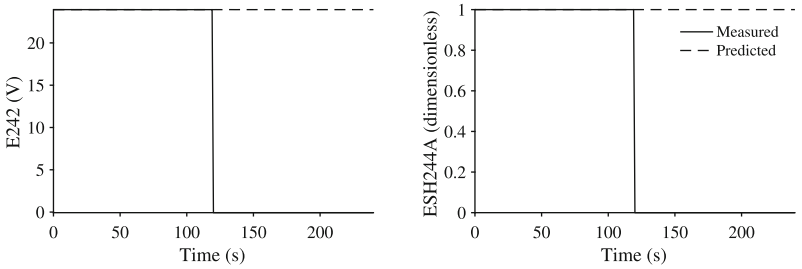

Fig. 8.5 Measured and predicted values for E242 and ESH244A for $EY244^{off}$

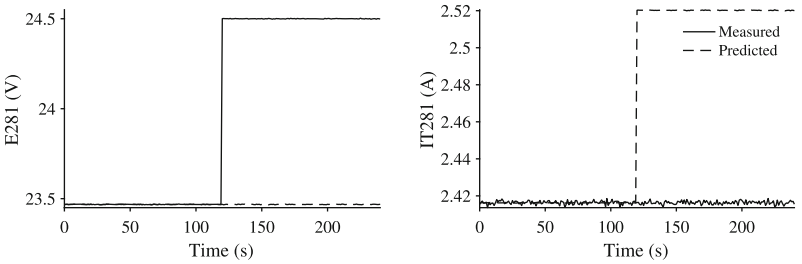

Fig. 8.6 Measured and predicted values for E281 and IT281 for b^{+}_{E281}

or ESH244A can produce a change in the relay position sensor, so given the previous information from E242, we can discover the true fault. In this case, no mode changes occurred near the time of fault detection, so the reasoning is simplified.

As a second example, consider the initial mode 11001, i.e., EY244, EY260, and EY281 are on, so power is sent only to the dc load. The fault, a positive bias in E281, occurs at 120.0 s. At 121.0 s, an increase in the residual of y_{E281} and decrease in the residual of y_{IT281} are detected (see Fig. 8.6). Considering first the increase in y_{E242}, the initial diagnoses are $\{b^{+}_{E281}, EY260^{off}\}$. Considering next the decrease in y_{IT281}, we can reduce the diagnoses to $\{b^{+}_{E281}\}$, which is the true fault. A mode change also occurs at 121.0 s, connecting the fan. This mode change does not change the modes of the submodels of interest so does not affect the reasoning process or produce new signatures.

A comprehensive set of experiments were performed in simulation to validate the approach. The initial system mode, fault, and sequence of mode changes were all selected randomly. For each experiment, we determined whether the true fault was found within the final set of diagnoses, and the diagnostic accuracy, computed as $1/|D|$, where D is the final set of diagnoses.

In 129 total experiments, the true fault was found 97.67% of the time. The average accuracy was 69.49%. Note that 100% accuracy is not expected since the system is not fully diagnosable in all modes, so in some cases there will be ambiguity based on qualitative fault signatures only. For the small percentage of the time in which the true fault was not found, this was due to false positives on some of the fault detectors

for the different residuals. In most of these cases, the true fault is found initially, but then eliminated because a false positive results in a signature inconsistent with the true fault. With additional tuning, the performance could potentially be improved.

Comparing discrete and parametric faults, when a discrete fault was the true fault, it was found 96.00% of the time, with an average accuracy of 67.67%. When a parametric fault was the true fault, it was found 98.08% of the time, with an average accuracy of 69.93%. Thus, performance was about equal for the two different fault types.

8.6 Conclusions

This chapter described a qualitative fault isolation approach for hybrid systems diagnosis. They key features are a compositional modeling approach and the use of structural model decomposition. Structural model decomposition plays a significant role in reducing the complexity of the hybrid systems diagnosis problem, by minimizing the local effects of faults to only a subset of residuals, reducing the number of mode changes to consider, and reducing the effects mode changes will have on the reasoning process.

The approach was demonstrated on a complex electrical power system, considering both parametric and discrete faults. Faults were quickly and correctly diagnosed, with some expected ambiguity due to the use of only qualitative information for diagnosis. This can be followed by quantitative fault identification to uniquely isolate the true fault.

Although only single faults are considered here, and all mode changes (except for faults) are assumed to be observable, the approach can be extended to handle multiple faults in the presence of unobservable mode changes. Preliminary work in this area has been described in the literature [1, 5, 19, 30]. Unobservable mode changes that are not tracked with nominal behavior will result in nonzero residuals, and thus appear as faults. Our framework can be extended to handle this case simply by considering fault signatures associated with these mode changes. As such, these mode changes would be identified.

Acknowledgements Matthew Daigle and Indranil Roychoudhury's work has been partially supported by the NASA SMART-NAS project in the Airspace Operations and Safety Program of the Aeronautics Mission Directorate. Anibal Begon's work has been funded by the Spanish MINECO DPI2013-45414-R grant.

References

1. Narasimhan, S., & Biswas, G. (2007). Model-based diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 37*(3), 348–361.

2. Cocquempot, V., El Mezyani, T., & Staroswiecki, M. (2004). Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference* (Vol. 2, pp. 1204–1212).
3. Sundström, C., Frisk, E., & Nielsen, L. (2014). Selecting and utilizing sequential residual generators in FDI applied to hybrid vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 44*(2), 172–185.
4. Koutsoukos, X., Kurien, J., & Zhao, F. (2003). Estimation of distributed hybrid systems using particle filtering methods. In *Hybrid Systems: Computation and Control (HSCC 2003). Lecture notes on computer science* (pp. 298–313). Berlin: Springer.
5. Hofbaur, M. W., & Williams, B. C. (2004). Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 34*(5), 2178–2191.
6. Benazera, E., & Travé-Massuyès, L. (2009). Set-theoretic estimation of hybrid system configurations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39*, 1277–1291.
7. Bayoudh, M., Travé-Massuyès, L., & Olive, X. (2008). Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis. In *18th European Conference on Artificial Intelligence* (pp. 219–223).
8. Bayoudh, M., Travé-Massuyès, L., & Olive, X. (2009). Diagnosis of a class of non linear hybrid systems by on-line instantiation of parameterized analytical redundancy relations. In *20th International Workshop on Principles of Diagnosis* (pp. 283–289).
9. Mosterman, P. J., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 29*(6), 554–565.
10. Daigle, M. J., Koutsoukos, X., & Biswas, G. (2009). A qualitative event-based approach to continuous systems diagnosis. *IEEE Transactions on Control Systems Technology, 17*(4), 780–793.
11. Narasimhan, S., & Brownston, L. (2007). HyDE: A general framework for stochastic and hybrid model-based diagnosis. In *Proceedings of the 18th International Workshop on Principles of Diagnosis* (pp. 186–193).
12. Trave-Massuyes, L., & Pons, R. (1997). Causal ordering for multiple mode systems. In *Proceedings of the Eleventh International Workshop on Qualitative Reasoning* (pp. 203–214).
13. Roychoudhury, I., Daigle, M., Bregon, A., & Pulido, B. (2013). A structural model decomposition framework for systems health management. In *Proceedings of the 2013 IEEE Aerospace Conference*.
14. Daigle, M., Bregon, A., & Roychoudhury, I. (2015). A structural model decomposition framework for hybrid systems diagnosis. In *Proceedings of the 26th International Workshop on Principles of Diagnosis*, Paris, France.
15. Bregon, A., Daigle, M., & Roychoudhury, I. (2016). Qualitative fault isolation of hybrid systems: A structural model decomposition-based approach. In *Third European Conference of the PHM Society 2016*.
16. Daigle, M., Bregon, A., & Roychoudhury, I. (2016). A qualitative fault isolation approach for parametric and discrete faults using structural model decomposition. In *Annual Conference of the Prognostics and Health Management Society 2016* (pp. 413–425).
17. Alonso, N. M., Bregon, A., Alonso-González, C. J., & Pulido, B. (2013). A common framework for fault diagnosis of parametric and discrete faults using possible conflicts. In *Advances in artificial intelligence* (pp. 239–249). Berlin: Springer.
18. Bregon, A., Narasimhan, S., Roychoudhury, I., Daigle, M., & Pulido, B. (2013). An efficient model-based diagnosis engine for hybrid systems using structural model decomposition. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*, 2013.
19. Daigle, M. (2008). *A Qualitative Event-Based Approach to Fault Diagnosis of Hybrid Systems*. PhD thesis, Vanderbilt University.

20. Daigle, M., Koutsoukos, X., & Biswas, G. (2010). An event-based approach to integrated parametric and discrete fault diagnosis in hybrid systems. *Transactions of the Institute of Measurement and Control, 32*(5), 487–510.
21. Poll, S., Patterson-Hine, A., Camisa, J., Garcia, D., Hall, D., Lee, C., et al. (2007). Advanced diagnostics and prognostics testbed. In *Proceedings of the 18th International Workshop on Principles of Diagnosis* (pp. 178–185).
22. Henzinger, T. A. (2000). *The theory of hybrid automata*. Berlin: Springer.
23. Mosterman, P. J., & Biswas, G. (2000). A comprehensive methodology for building hybrid models of physical systems. *Artificial Intelligence, 121*(1–2), 171–209.
24. Daigle, M., Roychoudhury, I., & Bregon A. (2015). Qualitative event-based diagnosis applied to a spacecraft electrical power distribution system. *Control Engineering Practice, 38*, 75–91.
25. Daigle, M., Roychoudhury, I., & Bregon, A. (2014). Qualitative event-based fault isolation under uncertain observations. In *Annual Conference of the Prognostics and Health Management Society 2014* (pp. 347–355).
26. Kurtoglu, T., Narasimhan, S., Poll, S., Garcia, D., Kuhn, L., de Kleer, J., et al. (2009). First international diagnosis competition – DXC'09. In *Proceedings of 20th International Workshop on Principles of Diagnosis* (pp. 383–396).
27. Poll, S., de Kleer, J., Abreau, R., Daigle, M., Feldman, A., Garcia, D., et al. (2011). Third international diagnostics competition – DXC'11. In *Proceedings of the 22nd International Workshop on Principles of Diagnosis* (pp. 267–278).
28. Sweet, A., Feldman, A., Narasimhan, S., Daigle, M., & Poll, S. (2013). Fourth international diagnostic competition – DXC'13. In *Proceedings of the 24th International Workshop on Principles of Diagnosis* (pp. 224–229).
29. Bregon, A., Biswas, G., Pulido, B., Alonso-González, C., & Khorasgani, H. (2014). A common framework for compilation techniques applied to diagnosis of linear dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 44*(7), 863–876.
30. Daigle, M., Bregon, A., Koutsoukos, X., Biswas, G., & Pulido, B. (2016). A qualitative event-based approach to multiple fault diagnosis in continuous systems using structural model decomposition. *Engineering Applications of Artificial Intelligence, 53*, 190–206.