# A Structural Model Decomposition Framework for Hybrid Systems Diagnosis

**Matthew Daigle**[1] and **Anibal Bregon**[2] and **Indranil Roychoudhury**[3]
[1]NASA Ames Research Center, Moffett Field, CA 94035, USA
e-mail: matthew.j.daigle@nasa.gov
[2]Department of Computer Science, University of Valladolid, Valladolid, 47011, Spain
e-mail: anibal@infor.uva.es
[3]Stinger Ghaffarian Technologies Inc., NASA Ames Research Center, Moffett Field, CA 94035, USA
e-mail: indranil.roychoudhury@nasa.gov

## Abstract

Nowadays, a large number of practical systems in aerospace and industrial environments are best represented as hybrid systems that consist of discrete modes of behavior, each defined by a set of continuous dynamics. These hybrid dynamics make the on-line fault diagnosis task very challenging. In this work, we present a new modeling and diagnosis framework for hybrid systems. Models are composed from sets of user-defined components using a compositional modeling approach. Submodels for residual generation are then generated for a given mode, and reconfigured efficiently when the mode changes. Efficient reconfiguration is established by exploiting causality information within the hybrid system models. The submodels can then be used for fault diagnosis based on residual generation and analysis. We demonstrate the efficient causality reassignment, submodel reconfiguration, and residual generation for fault diagnosis using an electrical circuit case study.

## 1 Introduction

Robust and efficient fault diagnosis plays an important role in ensuring the safe, correct, and efficient operation of complex engineering systems. Many engineering systems are modeled as *hybrid systems* that have both continuous and discrete-event dynamics, and for such systems, the complexity of fault diagnosis methodologies increases significantly. In this paper, we develop a new modeling framework and structural model decomposition approach that enable efficient online fault diagnosis of hybrid systems.

During the last few years, different proposals have been made for hybrid systems diagnosis, focusing on either hybrid modeling, such as hybrid automata [1–3], hybrid state estimation [4], or a combination of on-line state tracking and residual evaluation [5]. However, in all these cases, the proposed solutions involve modeling and pre-enumeration of the set of *all* possible system-level discrete modes, which grows exponentially with the number of switching components. Both steps are computationally very expensive or unfeasible for hybrid systems with complex interacting subsystems.

One solution to the mode pre-enumeration problem is to build hybrid system models in a *compositional* way, where discrete modes are defined at a local level (e.g., at the component level), in which 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 [6, 7]. 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.

In a system model, the effects of mode changes in individual components may force other components to reconfigure their computational structures, or *causality*, during the simulation process, which requires developing efficient online causality reassignment procedures. As an example of this kind of approach, Hybrid Bond Graphs (HBGs) [8] have been used by different authors [9, 10], and efficient causality reassignment has been developed previously for such models [11]. However, the main limitation of HBGs is that the set of possible components is restricted (e.g., resistors, capacitors, 0-junctions, etc.), with each component having to conform to a certain set of mathematical constraints, and modelers do not have the liberty to define and use their own components. Another example is that of [7], which uses a more general modeling framework, and tackles the causality reassignment problem from a graph-theoretic perspective.

In this work, we propose a compositional modeling approach for hybrid systems, where models are made up of sets of user-defined components. Here, a component is constructed by defining a set of discrete modes, with a different set of mathematical constraints describing the continuous dynamics in each mode. Then, we borrow ideas for efficient causality reassignment in HBGs [11], and propose algorithms for efficient causality assignment in our component-based models, extending and generalizing those from HBGs. We then apply structural model-decomposition [12] to compute minimal submodels for the initial mode of the system. These submodels are used for fault diagnosis based on residual generation and analysis. Based on efficient causality reassignment, submodels can be reconfigured upon mode changes efficiently. Using an electrical circuit as a case study, we demonstrate efficient causality reassignment and submodel reconfiguration and show that these submodels can correctly compute system outputs for residual generation in the presence of known mode changes.

The paper is organized as follows. Section 2 presents the modeling approach and introduces the case study. Section 3 presents the overall approach for hybrid systems fault diagnosis based on structural model decomposition. Section 4 develops the causality analysis and assignment algorithms. Section 5 presents the structural model decomposition ap-

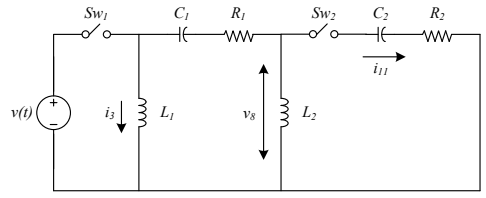



Figure 1: Electrical circuit running example.

proach. Section 6 describes efficient causality reassignment. Section 7 demonstrates the approach for the electrical case study. Section 8 reviews the related work and current approaches for hybrid systems fault diagnosis. Finally, Section 9 concludes the paper.

## 2 Compositional Hybrid Systems Modeling

We define hybrid system dynamics in a general compositional way, where the system is made up of a set of 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. Here, system-level modes are defined implicitly through the composition of the component-level modes. Because the number of system-level modes is exponential in the number of switching components, we want to avoid generating and reasoning over the system-level hybrid model, instead working directly with the component models.

To illustrate our proposal, throughout the paper we will use a circuit example, shown in Fig. 1. The components of the circuit are 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 $Sw_2$, as well as components for series and parallel connections. Sensors measure the current or voltage in different locations ($i_3$, $v_8$, and $i_{11}$, as indicated in Fig. 1). Because each switch has two modes (on and off), there are four total modes in the system.

In the following, we present the main details of our hybrid system modeling framework, which may be viewed as an extension of our modeling approach described in [12], extended with the notion of components, and with hybrid system dynamics.

### 2.1 System Modeling

At the basic level, the continuous dynamics of a component in each mode are modeled using a set of *variables* and a set of *constraints*. A constraint is defined as follows:

**Definition 1** (Constraint)**.** A constraint $c$ is a tuple $(\varepsilon_c, V_c)$, where $\varepsilon_c$ 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. A component is then defined as follows:

**Definition 2** (Component)**.** A *component* $\delta$ with $n$ discrete modes is a tuple $\delta = (V_\delta, \mathcal{C}_\delta)$, where $V_\delta$ is a set of variables and $\mathcal{C}_\delta$ is a set of constraints sets, where $\mathcal{C}_\delta$ is defined as $\mathcal{C}_\delta = \{C_\delta^1, C_\delta^2, \ldots, C_\delta^n\}$, with a constraint set, $C_\delta^m$, defined for each mode $m = \{1, \ldots, n\}$.

The components of the circuit are defined in Table 1 (first three columns).

**Example 1.** Consider the component $R_1$ ($\delta_6$). It has only a single mode with a single constraint $v_5 = i_5 * R_1$ over variables $\{v_5, i_5, R_1\}$.

**Example 2.** Consider the component $Sw_2$ ($\delta_{10}$). It has two modes: *on* and *off*. 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).

We can define a system model by composing components:

**Definition 3** (Model)**.** A *model* $\mathcal{M} = \{\delta_1, \delta_2, \ldots, \delta_d\}$ is a finite set of $d$ components for $d \in \mathbb{N}$.

**Example 3.** The model of the electrical system is made up of the components detailed in Table 1, i.e., $\mathcal{M} = \{\delta_1, \delta_2, \ldots, \delta_{15}\}$. For each component, the variables and constraints are defined for each component mode (third column).

Note that the set of variables for a model does not change with the mode, hence we need only a variable set in a component and not a set of variable sets as with constraints. The set of variables for a model, $V_\mathcal{M}$, is simply the union of all the component variable sets, i.e., for $d$ components, $V_\mathcal{M} = V_{\delta_1} \cup V_{\delta_2} \cup \ldots \cup V_{\delta_d}$. The interconnection structure of the model is captured using shared variables between components, i.e., we say that two components are connected if they share a variable, i.e., components $\delta_i$ and $\delta_j$ are connected if $V_{\delta_i} \cap V_{\delta_j} \neq \varnothing$. $V_\mathcal{M}$ consists of five disjoint sets, namely, the set of state variables, $X_\mathcal{M}$; the set of parameters, $\Theta_\mathcal{M}$; the set of inputs (variables not computed by any constraint), $U_\mathcal{M}$; the set of outputs (variables not used to compute any other variables), $Y_\mathcal{M}$; and the set of auxiliary variables, $A_\mathcal{M}$. Parameters, $\Theta_\mathcal{M}$, include explicit model parameters that are used in the model constraints (e.g., fault parameters). Auxiliary variables, $A_\mathcal{M}$, are additional variables that are algebraically related to the state, parameter, and input variables, and are used to simplify the structure of the equations.

**Example 4.** In the circuit model, we have $X_\mathcal{M} = \{i_3, v_6, i_8, v_{11}\}$, $\Theta_\mathcal{M} = \{L_1, R_1, C_1, L_2, R_2, C_2\}$, $U_\mathcal{M} = \{u_v\}$, and $Y_\mathcal{M} = \{i_3^*, i_{11}^*, v_8^*\}$. Remaining variables belong to $A_\mathcal{M}$. Here, the $^*$ superscript is used to denote a measured value of a physical variable, e.g., $i_3 \in X_\mathcal{M}$ is the current and $i_3^* \in Y_\mathcal{M}$ is the measured current. Since $i_3$ is used to compute other variables, like $i_2$, it cannot belong to $Y_\mathcal{M}$ and a separation of the variables is required. Connected components are known by shared variables, e.g., $R_1$ and Series Connection$_1$ are connected because they share $i_5$ and $v_5$.

The model constraints, $C_\mathcal{M}$, are a union of the component constraints over all modes, i.e., $C_\mathcal{M} = \mathcal{C}_{\delta_1} \cup \mathcal{C}_{\delta_2} \cup \ldots \cup \mathcal{C}_{\delta_d}$, where $\mathcal{C}_{\delta_i} = C_{\delta_i}^1 \cup C_{\delta_i}^2 \cup \ldots \cup C_{\delta_i}^n$ for $n$ modes. Constraints are exclusive to components, that is, a constraint $c \in C_\mathcal{M}$ belongs to exactly one $\mathcal{C}_\delta$ for $\delta \in \mathcal{M}$.

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 $\delta_1$, $\delta_2$, and $\delta_5$ use constraints of their mode 1, component $\delta_3$ use constraints of its mode 3, and component $\delta_4$ use constraints of its mode 2.

Table 1: Components of the electrical circuit.

| Component | Mode | Constraints | $\mathcal{A}^{[1\ 2]}$ | $\mathcal{A}^{[1\ 2]}_{i^*_3}$ | $\mathcal{A}^{[1\ 2]}_{v^*_8}$ | $\mathcal{A}^{[1\ 2]}_{i^*_{11}}$ | $\mathcal{A}^{[2\ 1]}$ | $\mathcal{A}^{[2\ 1]}_{i^*_3}$ | $\mathcal{A}^{[2\ 1]}_{v^*_8}$ | $\mathcal{A}^{[2\ 1]}_{i^*_{11}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\delta_1$: V | 1 | $v_1{=}u_v$ | $v_1$ | $v_1$ | | | $v_1$ | $v_1$ | $v_1$ | |
| $\delta_2$: $Sw_1$ | 1 | $i_1{=}0$ | $i_1$ | | | | | | | |
| | | $i_2{=}0$ | $i_2$ | $i_2$ | $i_2$ | $i_2$ | | | | |
| | 2 | $i_1{=}i_2$ | | | | | $i_1$ | | | |
| | | $v_1{=}v_2$ | | | | | $v_2$ | $v_2$ | $v_2$ | |
| $\delta_3$: Parallel Connection$_1$ | 1 | $v_2{=}v_3$ | $v_3$ | $v_3$ | | | $v_3$ | $v_3$ | | |
| | | $v_2{=}v_4$ | $v_2$ | $v_2$ | | | $v_4$ | | $v_4$ | |
| | | $i_2{=}i_3+i_4$ | $i_4$ | $i_4$ | $i_4$ | $i_4$ | $i_2$ | | | |
| $\delta_4$: $L_1$ | 1 | $\dot{i}_3{=}v_3/L_1$ | $\dot{i}_3$ | $\dot{i}_3$ | | | $\dot{i}_3$ | $\dot{i}_3$ | | |
| | | $i_3{=}\int_{t_0}^{t}\dot{i}_3$ | $i_3$ | $i_3$ | | | $i_3$ | $i_3$ | | |
| $\delta_5$: Series Connection$_1$ | 1 | $i_4{=}i_5$ | $i_5$ | $i_5$ | | | $i_5$ | | $i_5$ | |
| | | $i_4{=}i_6$ | $i_6$ | $i_6$ | | | $i_6$ | | $i_6$ | |
| | | $i_4{=}i_7$ | $i_7$ | | $i_7$ | $i_7$ | $i_4$ | | $i_4$ | |
| | | $v_4{=}v_5+v_6+v_7$ | $v_4$ | $v_4$ | | | $v_7$ | | $v_7$ | |
| $\delta_6$: $R_1$ | 1 | $v_5{=}i_5*R_1$ | $v_5$ | $v_5$ | | | $v_5$ | | $v_5$ | |
| $\delta_7$: $C_1$ | 1 | $\dot{v}_6{=}i_6/C_1$ | $\dot{v}_6$ | $\dot{v}_6$ | | | $\dot{v}_6$ | | $\dot{v}_6$ | |
| | | $v_6{=}\int_{t_0}^{t}\dot{v}_6$ | $v_6$ | $v_6$ | | | $v_6$ | | $v_6$ | |
| $\delta_8$: Parallel Connection$_2$ | 1 | $v_7{=}v_8$ | $v_8$ | $v_7$ | $v_8$ | | $v_8$ | | $v_8$ | |
| | | $v_7{=}v_9$ | $v_7$ | | $v_7$ | | $v_9$ | | | |
| | | $i_7{=}i_8+i_9$ | $i_9$ | | $i_9$ | $i_9$ | $i_7$ | | $i_7$ | |
| $\delta_9$: $L_2$ | 1 | $\dot{i}_8{=}v_8/L_2$ | $\dot{i}_8$ | | $\dot{i}_8$ | $\dot{i}_8$ | $\dot{i}_8$ | | $\dot{i}_8$ | |
| | | $i_8{=}\int_{t_0}^{t}\dot{i}_8$ | $i_8$ | | $i_8$ | $i_8$ | $i_8$ | | $i_8$ | |
| $\delta_{10}$: $Sw_2$ | 1 | $i_9{=}0$ | | | | | $i_9$ | | $i_9$ | |
| | | $i_{10}{=}0$ | | | | | $i_{10}$ | | | |
| | | $i_{11}{=}0$ | | | | | $i_{11}$ | | | $i_{11}$ |
| | 2 | $i_9{=}i_{10}$ | $i_{10}$ | | $i_{10}$ | | | | | |
| | | $i_9{=}i_{11}$ | $i_{11}$ | | | $i_{11}$ | | | | |
| | | $v_9{=}v_{10}+v_{11}$ | $v_9$ | | $v_9$ | | | | | |
| $\delta_{11}$: $R_2$ | 1 | $v_{10}{=}i_{10}*R_2$ | $v_{10}$ | | $v_{10}$ | | $v_{10}$ | | | |
| $\delta_{12}$: $C_2$ | 1 | $\dot{v}_{11}{=}i_{11}/C_1$ | $\dot{v}_{11}$ | | $\dot{v}_{11}$ | | $\dot{v}_{11}$ | | | |
| | | $v_{11}{=}\int_{t_0}^{t}\dot{v}_{11}$ | $v_{11}$ | | $v_{11}$ | | $v_{11}$ | | | |
| $\delta_{13}$: Current Sensor$_{11}$ | 1 | $i^*_{11}{=}i_{11}$ | $i^*_{11}$ | | $i_{11}$ | $i^*_{11}$ | $i^*_{11}$ | | | $i^*_{11}$ |
| $\delta_{14}$: Voltage Sensor$_8$ | 1 | $v^*_8{=}v_8$ | $v^*_8$ | $v_8$ | $v^*_8$ | $v_8$ | $v^*_8$ | | $v^*_8$ | |
| $\delta_{15}$: Current Sensor$_3$ | 1 | $i^*_3{=}i_3$ | $i^*_3$ | $i^*_3$ | $i_3$ | $i_3$ | $i^*_3$ | $i^*_3$ | | |

For shorthand, we will refer to the modes only of the components with multiple modes. So, for the circuit, we will refer only to components $\delta_2$ and $\delta_{10}$, and we will have 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. The state transitions may be attributed to controlled or autonomous events. However, for the purposes of this paper, we view the switching behavior 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 [1, 8].

## 2.2 Causality

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 $c$. This is done by defining which $v \in V_c$ is the dependent variable in equation $\varepsilon_c$.

**Definition 4** (Causal Assignment)**.** A *causal assignment* $\alpha_c$ 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 $\varepsilon_c$. We use $V_c^{in}$ to denote the independent variables in the constraint, where $V_c^{in} = V_c - \{v_c^{out}\}$.

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 we have noninvertible nonlinear constraints. Also, if we assume integral causality, then state variables must always be computed via integration, and so the derivative causality is not allowed. Further, when placed in the context of a model, additional causalities may not be applicable, because the causal assignments of other constraints may limit the potential causal assignments. To denote this concept, we use $\mathbb{A}_c$ to refer to the set of permissible causal assignments of a constraint $c$.

For a given mode, we have the set of (specific) causal assignments over the entire model in its mode, denoted using $\mathcal{A}^{\mathbf{m}}$. So, some $\alpha \in \mathcal{A}^{\mathbf{m}}$ would refer to the causal assignment of some constraint in some component of the model in its correct mode. The consistency of the causal assignments $\mathcal{A}^{\mathbf{m}}$ is defined as follows,

**Definition 5** (Consistent Causal Assignments)**.** Given a mode $m$, we say that a set of causal assignments $\mathcal{A}^{\mathbf{m}}$, for a model $\mathcal{M}$ is *consistent* if (*i*) for all $v \in U_{\mathcal{M}} \cup \Theta_{\mathcal{M}}$, $\mathcal{A}^{\mathbf{m}}$ does not contain any $\alpha$ such that $\alpha = (c, v)$, i.e., input or parameter variables cannot be the dependent variables in the causal assignment; (*ii*) for all $v \in Y_{\mathcal{M}}$, $\mathcal{A}^{\mathbf{m}}$ does not contain any $\alpha = (c, v_c^{out})$ where $v \in V_c^{in}$, i.e., an output variable can only be used as the dependent variable; and (*iii*) for all $v \in V_{\mathcal{M}} - U_{\mathcal{M}} - \Theta_{\mathcal{M}}$, $\mathcal{A}^{\mathbf{m}}$ contains exactly one $\alpha = (c, v)$, i.e., every variable that is not an input or

parameter is computed by only one (causal) constraint.

With causality information, we can efficiently derive a set of submodels for residual generation [12].

## 3 Hybrid Systems Diagnosis Approach

We propose a hybrid systems diagnosis approach based on structural model decomposition. In this approach, we generate submodels for the purpose of computing residuals. Residuals can then be used for diagnosis.

For hybrid systems, however, the problem is that these submodels may change as the result of a mode change. That is, we may obtain two different submodels when decomposing the model in two different modes. There are two approaches to this problem. One is to find a set of submodels that work for all modes, and can be easily reconfigured by executing only local mode changes within the submodels [10]. This approach requires the least online effort, with some offline effort in finding these submodels, which exist only in limited cases. The other approach is to generate submodels for the current mode, and when a mode change occurs, reconfigure the submodels to be consistent with the new system mode. This is the approach we develop in this paper.

In order to execute this type of approach, however, we must be able to efficiently reconfigure submodels online. In order to do this, we take advantage of causality in two ways. First, we perform an offline model analysis to determine which causalities of the hybrid system model are not permissible, i.e., they will never be used in any mode of the system (determine $\mathbb{A}_{\mathcal{M}}$ for a model). Second, we use an efficient causality reassignment algorithm, so that the causality of a hybrid systems model is updated incrementally when a mode changes (given $\mathcal{A}$ for the previous mode, compute it for the new mode). Since causal changes usually only propagate in a local area in the model, causality does not need to be reassigned at the global model level. Together, these algorithms reduce the number of potential causalities to search within the model decomposition algorithm and allow efficient submodel reconfiguration.

## 4 Causality Assignment

In order to compute minimal submodels for residual generation, we need a model $\mathcal{M}$ with a valid causal assignment $\mathcal{A}^{\mathbf{m}}$. As described in Section 2, causality assignment can only be defined for a given mode. However, there are some causal assignments that are independent of the system mode, i.e., they are valid for all system modes. We capture this through the notion of permissible causal assignments, introduced as $\mathbb{A}_{\mathcal{M}}$ in Section 2.

Given a model with a number of modes, some constraints will always have the same causal assignment in all modes, and we say these constraints are in *fixed causality*.

**Definition 6** (Fixed Causality)**.** A constraint $c_\delta$ is in *fixed causality* if (*i*) component $\delta$ has only a single mode, i.e., $|\mathcal{C}_\delta| = 1$, and (*ii*) for $c_\delta$ in the single $C \in \mathcal{C}_\delta$, it always has the same causal assignment in all system modes.

If a constraint is in fixed causality, then $|\mathbb{A}_c| = 1$, i.e., there is only one permissible causal assignment. For example, if we make the integral causality assumption, then constraints computing state variables will always be in the integral causality, and thus they are in fixed causality.

Additionally, when the constraint is viewed in the context of the model, the concept of fixed causality can be propagated from one constraint to the related constraints (those sharing a variable with the fixed causality constraint). This will help to reduce the number of permissible causal assignments. For example, if we again assume integral causality, then any constraint involving a state variable cannot be in a causal assignment where the state variable is the dependent/output variable, because the integration constraint is the one that must compute it. For such a constraint, $1 < |\mathbb{A}_c| < |V_c|$.

Given a system model and a set of outputs, Algorithm 1 searches over the model constraints to reduce the set of permissible causal constraints based on system-level information.[1] First, it determines which constraints are mode-variant, i.e, they can appear/disappear from the model depending on the mode (so belong to components with multiple modes), and which are mode-invariant, i.e., they are present in all system modes (so belong to components with a single mode). It is only the mode-invariant constraints for which causal assignments can be removed. We then construct a queue of variables from which to propagate. This queue contains the inputs and parameters (which must always be independent/input variables in constraints), and the outputs (which must always be dependent/output variables in constraints). We create a variable set $V$ that refers to the variables that are resolved, i.e., either they are inputs/parameters or there is a constraint with a single causal assignment that will compute the variable. So, $V$ is initially set to include $U_{\mathcal{M}}$ and $\Theta_{\mathcal{M}}$. Further, for any mode-invariant constraints that only has a single causal assignment, the output variable is added to $V$, and all variables of the constraint added to the queue.

The main idea is to analyze the causality restrictions imposed by variables in the queue, which will be propagated throughout the model. While the queue is nonempty, we pop a variable $v$ off the queue. We then count the number of constraints involving $v$ that have no set causal assignment yet, including constraints that are both mode-variant and mode-invariant. We then go through all mode-invariant constraints involving $v$, and remove causal assignments that will never be possible. There are three conditions in which this holds: a causal assignment is not possible in any system mode if (*i*) the output variable is already computed by another constraint, or is an input/parameter (i.e., in $V$), (*ii*) any of the input variables are in the model outputs (i.e., in $Y$), or (*iii*) $v$ is not yet computed by any constraint (i.e., not in $V$), there is only one noncausal constraint involving $v$ remaining, and $v$ is not the output in this causality (in this case, $v$ needs to be computed by some constraint and there is only one option left, so this constraint must only be in the causality computing $v$). These causal assignments are removed. If only one is left, then we add the output for that causal assignment to $V$, and add the constraint's variables to the queue. The algorithm stops when causalities can no longer be removed, i.e., there are not enough restrictions imposed by the current permissible causalities to reduce $\mathbb{A}_{\mathcal{M}}$ further.

**Example 6.** For the circuit, we assume integral causality, so all constraints with the state variables are limited to causal assignments in which the states are computed via integration. Further, the constraint with $u_V$ is also fixed so that $u_V$ is the independent variable. For any specified outputs, $\mathbb{A}_{\mathcal{M}}$ is also

[1] For structural model decomposition, some output variables may become input variables and so the causal assignments permitting that must be retained. Therefore, the algorithm only reduces the permissible set of causal assignments for a given set of outputs $Y \subseteq Y_{\mathcal{M}}$.

**Algorithm 1** $\mathbb{A}_{\mathcal{M}} \leftarrow$ `ReduceCausality`$(\mathcal{M}, \mathbb{A}_{\mathcal{M}}, Y)$

$C_{\text{invariant}} \leftarrow \varnothing$
$C_{\text{variant}} \leftarrow \varnothing$
**for all** $\delta \in \mathcal{M}$ **do**
  **if** $|\mathcal{C}_\delta| = 1$ **then**
    $C_{\text{invariant}} \leftarrow C_{\text{invariant}} \cup C_\delta^1$
  **else**
    $C_{\text{variant}} \leftarrow C_{\text{variant}} \cup \left( \bigcup_{C \in \mathcal{C}_\delta} C \right)$
$Q \leftarrow U_{\mathcal{M}} \cup \Theta_{\mathcal{M}} \cup Y$
$V \leftarrow U_{\mathcal{M}} \cup \Theta_{\mathcal{M}}$
**for all** $c \in C_{\text{invariant}}$ **do**
  **if** $|\mathbb{A}_c| = 1$ **then**
    $(c, v) \leftarrow \mathbb{A}_c(1)$
    $Q \leftarrow Q \cup V_c$
    $V \leftarrow V \cup v$
**while** $|Q| > 0$ **do**
  $v \leftarrow \texttt{pop}(Q)$
  $n_{\text{noncausal}} \leftarrow 0$
  **for all** $c \in C_{\text{invariant}}(v)$ **do**
    **if** $|\mathbb{A}_c| > 1$ **or** $(|\mathbb{A}_c| = 1$ **and** $v_{\mathbb{A}_c(1)} \notin V)$ **then**
      $n_{\text{noncausal}} \leftarrow n_{\text{noncausal}} + 1$
  **for all** $c \in C_{\text{variant}}(v)$ **do**
    $n_{\text{noncausal}} \leftarrow n_{\text{noncausal}} + 1$
  **for all** $c \in C_{\text{invariant}}(v)$ **do**
    **if** $|\mathbb{A}_c| > 1$ **or** $(|\mathbb{A}_c| = 1$ **and** $v_{\mathbb{A}_c(1)} \notin V)$ **then**
      **for all** $(c', v') \in \mathbb{A}_c$ **do**
        **if** $v' \in V$ **then**
          $\mathbb{A}_c \leftarrow \mathbb{A}_c - (c', v')$
        **if** $(V_c - \{v\}) \cap Y \neq \varnothing$ **then**
          $\mathbb{A}_c \leftarrow \mathbb{A}_c - (c', v')$
        **if** $n_{\text{noncausal}} = 1$ **and** $v' \notin V$ **and** $v' \neq v$ **then**
          $\mathbb{A}_c \leftarrow \mathbb{A}_c - (c', v')$
      **if** $|\mathbb{A}_c| = 1$ **then**
        $(c', v') \leftarrow \mathbb{A}_c(1)$
        $Q \leftarrow Q \cup (V_{c'} - V)$
        $V \leftarrow V \cup \{v'\}$

reduced so that they can appear only as dependent variables.

With $\mathbb{A}_{\mathcal{M}}$ defined, we can perform causality assignment for a given mode, $\mathbf{m}$. Because $\mathbb{A}_{\mathcal{M}}$ was reduced as much as possible, causality assignment (and, later, reassignment) will be more efficient than otherwise. Algorithm 2 describes the causality assignment process for a model given a mode. Here, the model is assumed to not have an initial causal assignment. Causal assignment works by propagating causal restrictions throughout the model. The process starts at inputs, which must always be independent variables in constraints; outputs, which must always be the dependent variables in constraints; and variables for involved in fixed causality constraints. 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 (this is a familiar concept within bond graphs) [13].

Specifically, the algorithm works as follows. Similar to Algorithm 1, we keep a 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 $U$ and $\Theta$, because these variables are not to be computed by any constraint. $Q$ is set to $U$, $\Theta$, and $Y$, since the causality of

[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.

**Algorithm 2** $\mathcal{A} \leftarrow$ `AssignCausality`$(\mathcal{M}, \mathbf{m}, \mathbb{A})$

$\mathcal{A} \leftarrow \varnothing$
$V \leftarrow U_{\mathcal{M}} \cup \Theta_{\mathcal{M}}$
$Q \leftarrow U_{\mathcal{M}} \cup \Theta_{\mathcal{M}} \cup Y_{\mathcal{M}}$
**for all** $c \in C_{\mathcal{M}\mathbf{m}}$ **do**
  **if** $|\mathbb{A}_c| = 1$ **then**
    $(c, v) \leftarrow \mathbb{A}_c(1)$
    $Q \leftarrow Q \cup v$
**while** $|Q| > 0$ **do**
  $v \leftarrow \texttt{pop}(Q)$
  **for all** $c \in C_{\mathcal{M}\mathbf{m}}(v)$ **do**
    **if** $c \notin \{c : (c, v) \in \mathcal{A}\}$ **then**
      $\alpha^* \leftarrow \varnothing$
      **for all** $\alpha \in \mathbb{A}_c$ **do**
        **if** $V_c - \{v_{\alpha^*}\} \cup V \neq \varnothing$ **then**
          $\alpha^* \leftarrow \alpha$
        **else if** $\alpha_v \in Y$ **then**
          $\alpha^* \leftarrow \alpha$
        **else if** $v_{\alpha^*} = v$ **and** $|C_{\mathcal{M}\mathbf{m}}(v)| - |\{c' : (c', v') \in \mathcal{A} \wedge v \in v_c\}| = 1$ **then**
          $\alpha^* \leftarrow \alpha$
      **if** $\alpha^* \neq \varnothing$ **then**
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{\alpha^*\}$
        $Q \leftarrow Q \cup (V_c - V)$
        $V \leftarrow V \cup \{v_{\alpha^*}\}$

constraints is restricted to $U$ and $\Theta$ 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 $\mathcal{A}$.

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 $\mathcal{A}$), we go through all permissible causal assignments, and determine if the causality is forced into one particular causal assignment, $\alpha^*$. 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 $v$.

**Example 7.** Consider the mode $\mathbf{m} = [1\ 2]$. Here, $\mathcal{A}^{[1\ 2]}$ is given in column 4 of Table 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 8.** Consider the mode $\mathbf{m} = [2\ 1]$. Here, $\mathcal{A}^{[2\ 1]}$ is given in column 8 of Table 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. In changing from one mode to another, an efficient causality reassignment should be able to determine which constraints need to change causality, and do the work for only that portion of the model.[3] Causal assignments that do not change from mode to mode are in fixed causality and found by Algorithm 1.

[3]Note that this particular circuit was carefully chosen so that causality does propagate across much of the circuit, in order to demonstrate the causality reassignment algorithm.

## 5 Structural Model Decomposition

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 our previous work [12]. 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. Additional information and the pseudocode are provided in [12].

In the context of residual generation, we set the local output set to a single measured value, and the local inputs to all other measured values and the (known) system inputs. That is, we exploit the analytical redundancy provided by the sensors in order to find minimal submodels to compute estimated values of sensor outputs. In this framework, we consider one submodel per sensor, each producing estimated values for that sensor.

Assuming that the set of sensors does not change from mode to mode, then for a hybrid system we have one submodel for each sensor.[4] However, since the set of constraints changes from mode to mode, the result of the `GenerateSubmodel` algorithm will also change. When a mode changes, we first reassign causality to the model for the new mode. Then, we generate new updated submodels for that mode using `GenerateSubmodel`. In order to reduce the work performed by this algorithm when a mode changes, we use an efficient causality reassignment algorithm. That, coupled with the reduced set $\mathbb{A}_{\mathcal{M}}$, significantly reduces the work of the algorithm compared to a naive approach, where the submodels are completely regenerated for a new mode. Additionally, when the system transitions to a new mode, the causal assignments for the previous mode can be stored, so that when the system changes to a mode that has already been visited, it just takes the causal assignments that were stored previously. Similarly, submodels generated in previously visited modes can be saved and reused when the mode reappears.

**Example 9.** The causal assignments for the submodels in the different modes are shown in Table 1. For example, consider the submodel for $i_{11}^*$ in $\mathbf{m} = [2\ 1]$. Here, $i_{11}$ is zero, since $\mathrm{Sw}_2$ is off, and therefore we have just two constraints needed to compute $i_{11}^*$. In mode $\mathbf{m} = [1\ 2]$, $i_3^*$ can be computed using 16 constraints, where $v_8^*$ is used as a local input to the submodel.

Note that a submodel for an output may have different states in two different modes (e.g., in moving from $\mathbf{m} = [2\ 1]$ to $\mathbf{m} = [1\ 2]$, the $i_3^*$ submodel adds state $v_6$). In order to continue tracking, new states must be initialized. For the purposes of this paper, we assume that in any one system mode, all states are included in at least one submodel.[5] Therefore, a submodel that gets a state added in a new mode can initialize using the estimated value from another submodel in the previous mode.

**Algorithm 3** $\mathcal{A}^{\mathbf{m}'} \leftarrow$ `ReassignCausality`$(\mathcal{M}, \mathbf{m}, \mathcal{A}^{\mathbf{m}}, \mathbb{A})$

```
1:  A^m' ← ∅
2:  for all (c, v) ∈ A^m do
3:      if c ∈ C_{M^m} then
4:          A^m' ← A^m' ∪ A_c^m
5:  V ← ∅
6:  Q ← ∅
7:  for all δ ∈ M where m_δ ≠ m'_δ do
8:      Q ← Q ∪ V_δ
9:  while |Q| > 0 do
10:     v ← pop(Q)
11:     for all c ∈ C_{M^m}(v) do
12:         if c ∉ {c : (c, v) ∈ A^m'} then
13:             α* ← ∅
14:             for all α ∈ A_c do
15:                 if V_c − {v_α*} ∪ V ≠ ∅ then
16:                     α* ← α
17:                 else if α_v ∈ Y then
18:                     α* ← α
19:                 else if v_α* = v and |C_{M^m}(v)| − |{c' : (c', v') ∈ A ∧ v ∈
                        v_c}| = 1 then
20:                     α* ← α
21:             if α* ≠ ∅ then
22:                 if ∃α ∈ A^m' where v_α = v*_α then
23:                     A^m' ← A^m' − {α*}
24:                     Q ← Q ∪ (V_{α*_c} − V)
25:                 A^m' ← A^m' ∪ {α*}
26:                 Q ← Q ∪ (V_c − V)
27:                 V ← V ∪ {v_α*}
28:         else if (V_c − V = v then
29:             V ← V ∪ {v}
30:             Q ← Q ∪ {v}
```

## 6 Online Causality Reassignment

As we mentioned before, from the initial mode in the system with a valid set of causal assignments, we compute minimal submodels. However, when the system transitions to a different mode, any submodel containing constraints of a switching component will no longer be consistent, and must be recomputed. In order to do this, we need to know the causal assignments for the new mode. We can reassign causality in an efficient incremental process to avoid having to reassign causality to the whole model, as causal changes typically propagate only to a small local area in the model [11].

Algorithm 3 presents the causality reassignment procedure. The main ideas are based on the hybrid sequential causality assignment procedure (HSCAP) developed for hybrid bond graphs in [11]. In our more general modeling framework, we find that similar ideas apply. Essentially, we start with a causal model in a given mode. We then switch to a different mode, so for the switching components we have a new set of constraints in the model. We need to find causal assignments for these constraints. It is likely that some of the necessary causal assignments will conflict with causal assignments from the old mode, therefore, we have to resolve the conflict and propagate the change. The change will propagate only as far as it needs to in order to obtain a valid causal assignment for the model in the new mode. Here, propagation stops along a computational path when a new causal assignment does not conflict with a previous assignment.

[4] By assuming that the sensor set does not change, we mean only that sensors are not added/removed to/from the physical system upon a mode change. They are still allowed to be connected/disconnected, but still appear in the system model even when disconnected. For example, if a disconnected sensor outputs 0, then that needs to still be in the model.

[5] If this is not the case, then a state is not observable in some mode. Estimation techniques to handle that situation are outside the scope of this paper.

The algorithm works similarly to Algorithm 2. It maintains a queue of variables to inspect and a set $V$ of variables that are known to be computed in the causality for the new mode (so includes only variables from new assignments or confirmed assignments made in the new mode). In this case, we initialize the queue only to variables involved in the constraints of the switching components. If no components are switching, the queue will be empty and no work will be done. The main idea is that the required causal changes from the variables placed in the queue will, on average, be limited to a very small area. The causal assignments for the new mode are initialized to those for the previous mode, for any constraints that still exist in the new mode. Some of these may conflict with the new mode and will be removed and replaced with different assignments.

As in all the other causality algorithms, we go through the queue and propagate the restrictions we find on causality. We pop a variable off the queue, and look at all involved constraints. If the constraint is not causal, then we need to assign causality. We do the same analysis as before to find if a causality is forced, but checking things only with respect to $V$ that includes only variables with confirmed causal assignments computing it in the new mode. If we find a constraint that is forced into a particular causal assignment for the new mode, we make the assignment. If it conflicts with one already in the set of causal assignments (copied from the old mode), then we remove the old assignment and add the new one, adding the involved variables to the queue so that changes are propagated.

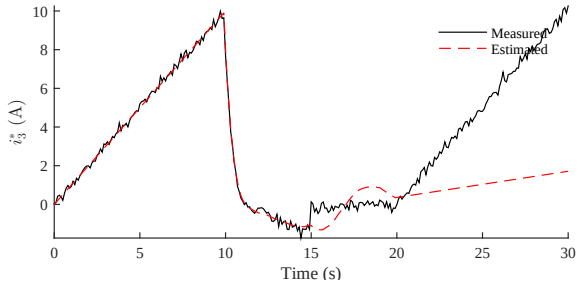
## 7 Demonstration of Approach

For the circuit example, we consider two modes: one where $Sw_1$ is on and $Sw_2$ is off (i.e., $\mathbf{m} = [2\ 1]$), and one where $Sw_1$ is off and $Sw_2$ is on (i.e., $\mathbf{m} = [1\ 2]$). We consider a scenario in which to demonstrate the approach where the system starts in $\mathbf{m} = [2\ 1]$, switches to $\mathbf{m} = [1\ 2]$ at $t = 10$ s, and switches back to $\mathbf{m} = [2\ 1]$ at $t = 20$ s. Additionally, at $t = 15$ s, a fault is injected, specifically, an increase in $R_1$.

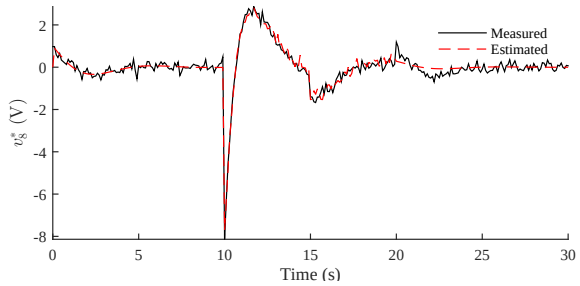
Fig. 2 shows the measured and submodel-estimated values for the sensors. Up through the first mode change, the outputs are correctly tracked by the submodels. At the first mode change at 10 s, the submodels reconfigure and track correctly up to 15 s, when the fault is injected, and a discrepancy is observed in the $i_3^*$ submodel. Specifically, the current increases above what is expected. The other submodels in this mode are independent of the fault, and so continue to track correctly. When the second mode change occurs, $i_{11}^*$ can still be tracked correctly, since its estimation remains independent of the fault. However, we now see a discrepancy in $v_8^*$, as the measurement increases above what is expected. This transient occurs because we switch from a mode in which the submodel is independent of the fault to one where it is dependent on the fault. Fault isolation can be performed by using the information that in $\mathbf{m} = [1\ 2]$, an increase in $R_1$ would produce an increase in the residual for $i_3^*$, and in $\mathbf{m} = [2\ 1]$, it would produce an increase also in the $v_8^*$ residual.

## 8 Related Work

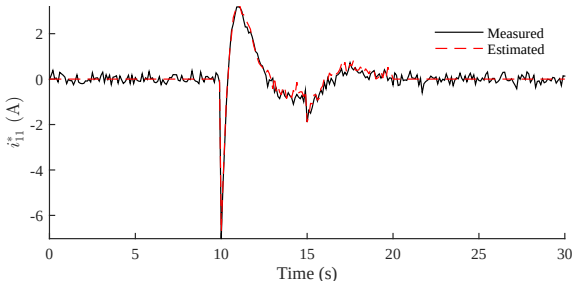
Modeling and diagnosis for hybrid systems have been an important focus of study for researchers from both the FDI and DX communities during the last 15 years. In the FDI community, several hybrid system diagnosis approaches have been



(a) $i_3^*$.



(b) $v_8^*$



(c) $i_{11}^*$

Figure 2: Measured and estimated values with an increase in $R_1$ at $t = 15$ s.

developed. In [14], parameterized ARRs are used. However, the approach is not suitable for systems with high nonlinearities or a large set of modes. A different approach [15], but uses purely discrete models.

In the DX community, some approaches have used different kind of automata to model the complete set of modes and transitions between them. In those cases, the main research topic has been hybrid system state estimation, which has has been done using probabilistic (e.g., some kind of filter [16] or hybrid automata [4]) or set-theoric approaches [5].

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 [3], or parameterized ARRs for the complete set of modes [17]). Nevertheless, one of the main difficulties regarding state estimation using these techniques is the need to pre-enumerate the set of possible system-level modes and mode transitions, which is difficult for complex systems. We avoid this problem by using a compositional approach.

Regarding hybrid systems modeling, there are several proposals. For HBGs [8, 18], there are two main approaches: those that use switching elements with fixed causality [18–20], and those who use ideal switching elements that change causality [8]. The advantages of the latter are that the modeling of hybrid systems is done through a special kind of

hybrid component (which avoid the mode pre-enumeration in the system), and also changes are handled in a very efficient way [11]. Finally, in [10] the HBGs are used to compute minimal submodels (Hybrid Possible Conflicts, HPCs) similar to the minimal submodels presented in this paper. HPCs can track hybrid systems behavior, efficiently changing online for each mode the PC simulation model, by using block diagrams as in [11], and performing diagnosis without pre-enumerating the set of modes in the system. However, HPCs rely on HBG modeling and do not provide a generalized framework for hybrid systems.

## 9 Conclusions

In this work, we have developed a compositional modeling framework for hybrid systems. Using computational causality, we developed efficient causality assignment algorithms. Given this causal information, submodels computed using structural model decomposition can be computed and reconfigured efficiently. The approach was demonstrated with a circuit system. In future work, we will further develop the hybrid systems diagnosis approach for the single and multiple fault cases, and we will approach the diagnosis task in a distributed manner. The assumption of one submodel per sensor can also be dropped, using the extended framework developed in [21, 22].

## Acknowledgments

This work has been funded by the Spanish MINECO DPI2013-45414-R grant and the NASA SMART-NAS project in the Airspace Operations and Safety Program of the Aeronautics Mission Directorate.

## References

[1] T. A. Henzinger. *The theory of hybrid automata*. Springer, 2000.

[2] T. Rienmüller, M. Bayoudh, M.W. Hofbaur, and L. Travé-Massuyès. Hybrid Estimation through Synergic Mode-Set Focusing. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1480–1485, Barcelona, Spain, 2009.

[3] M. Bayoudh, L. Travé-Massuyès, and X. Olive. Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis. In *18th European Conf. on Artificial Intel.*, pages 219–223, 2008.

[4] M.W. Hofbaur and B.C. Williams. Hybrid estimation of complex systems. *IEEE Trans. on Sys., Man, and Cyber, Part B: Cyber.*, 34(5):2178–2191, 2004.

[5] E. Benazera and L. Travé-Massuyès. Set-theoretic estimation of hybrid system configurations. *Trans. Sys. Man Cyber. Part B*, 39:1277–1291, October 2009.

[6] S. Narasimhan and L. Brownston. HyDE: A General Framework for Stochastic and Hybrid Model-based Diagnosis. In *Proc. of the 18th Int. WS. on Principles of Diagnosis*, pages 186–193, May 2007.

[7] L. Trave-Massuyes and R. Pons. Causal ordering for multiple mode systems. In *Proceedings of the Eleventh International Workshop on Qualitative Reasoning*, pages 203–214, 1997.

[8] P.J. Mosterman and G. Biswas. A comprehensive methodology for building hybrid models of physical systems. *Artificial Intel.*, 121(1-2):171 – 209, 2000.

[9] S. Narasimhan and G. Biswas. Model-Based Diagnosis of Hybrid Systems. *IEEE Trans. Syst. Man. Cy. Part A*, 37(3):348–361, May 2007.

[10] A. Bregon, C. Alonso, G. Biswas, B. Pulido, and N. Moya. Hybrid systems fault diagnosis with possible conflicts. In *Proceedings of the 22nd International Workshop on Principles of Diagnosis*, pages 195–202, Murnau, Germany, October 2011.

[11] I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos. Efficient simulation of hybrid systems: A hybrid bond graph approach. *SIMULATION: Transactions of the Society for Modeling and Simulation International*, 87(6):467–498, June 2011.

[12] I. Roychoudhury, M. Daigle, A. Bregon, and B. Pulido. A structural model decomposition framework for systems health management. In *Proceedings of the 2013 IEEE Aerospace Conference*, March 2013.

[13] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg. *Systems Dynamics: Modeling and Simulation of Mechatronic Systems*. John Wiley & Sons, Inc., NY, 2000.

[14] V. Cocquempot, T. El Mezyani, and M. Staroswiecki. Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference*, volume 2, pages 1204–1212, July 2004.

[15] J. Lunze. Diagnosis of quantised systems by means of timed discrete-event representations. In *Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control*, HSCC '00, pages 258–271, London, UK, 2000. Springer-Verlag.

[16] X. Koutsoukos, J. Kurien, and F. Zhao. Estimation of distributed hybrid systems using particle filtering methods. In *In Hybrid Systems: Computation and Control (HSCC 2003). Springer Verlag Lecture Notes on Computer Science*, pages 298–313. Springer, 2003.

[17] M. Bayoudh, L. Travé-Massuyès, and X. Olive. 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*, pages 283–289, 2009.

[18] W. Borutzky. Representing discontinuities by means of sinks of fixed causality. In F.E. Cellier and J.J. Granda, editors, *Proc. of the Int. Conf. on Bond Graph Modeling*, pages 65–72, 1995.

[19] M. Delgado and H. Sira-Ramirez. Modeling and simulation of switch regulated dc-to-dc power converters of the boost type. In *IEEE Int. Conf. on Devices, Circuits and Systems*, pages 84–88, December 1995.

[20] P.J. Gawthrop. Hybrid Bond Graphs Using Switched I and C Components. CSC report 97005, Centre for Sys. and Control, Faculty of Eng., Glasgow, U.K., 1997.

[21] A. Bregon, M. Daigle, and I. Roychoudhury. An integrated framework for distributed diagnosis of process and sensor faults. In *2015 IEEE Aerospace Conf.*, 2015.

[22] M. Daigle, I. Roychoudhury, and A. Bregon. Diagnosability-based sensor placement through structural model decomposition. In *Second Euro. Conf. of the PHM Society 2014*, pages 33–46, 2014.