A QUALITATIVE EVENT-BASED APPROACH TO FAULT DIAGNOSIS OF HYBRID SYSTEMS

By

Matthew J. Daigle

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in
Computer Science

May, 2008
Nashville, Tennessee

Approved:
Professor Xenofon Koutsoukos
Professor Gautam Biswas
Professor Gabor Karsai
Professor Sankaran Mahadevan
Professor Nilanjan Sarkar

PREFACE

Fault diagnosis is crucial for ensuring the safe operation of complex engineering systems. Many present-day systems combine physical and computational processes, and are best modeled as hybrid systems, where the dynamic behavior combines continuous evolution interspersed with discrete configuration changes. Due to the complexity of such modern engineering systems, formal methods are required for reliable and correct design, analysis, and implementation of hybrid system diagnosers.

This dissertation presents a systematic, model-based approach to event-based diagnosis of hybrid systems based on qualitative abstractions of deviations from nominal behavior. The primary contributions of this work center on (*i*) incorporating relative measurement orderings into fault isolation for continuous and hybrid systems, which describe predicted temporal orderings of measurement deviations, (*ii*) providing algorithms for event-based diagnosis of single and multiple faults, (*iii*) developing an integrated framework for diagnosis of parametric, sensor, and discrete, i.e., switching faults in hybrid systems, and (*iv*) developing and implementing an efficient event-based diagnosis framework for continuous and hybrid systems that enables automatic design of event-based diagnosers and establishes notions of diagnosability for continuous and hybrid systems.

The effectiveness of the approach is demonstrated on two practical systems. First, the single fault diagnosis method for continuous systems is applied in a distributed fashion to formations of mobile robots. The results include a formal diagnosability analysis, scalability results, and experiments performed on a formation of robots. Second, the approach developed for hybrid systems diagnosis is applied to the Advanced Diagnostics and Prognostics Testbed, which is a complex electrical distribution system for spacecraft and aircraft applications. The results focus on a subset of the testbed, and include a diagnosability analysis, experiments from the actual testbed, and detailed simulation experiments that examine the performance of the diagnosis algorithms for different fault magnitudes and noise levels.

ACKNOWLEDGMENTS

In the completion of my research and this dissertation, I owe thanks to my advisors, Professors Xenofon Koutsoukos and Gautam Biswas, whose comments and critiques helped improve the quality and presentation of the work. I would also like to thank the remaining members of the committee, Professors Gabor Karsai, Sankaran Mahadevan, and Nilanjan Sarkar, for their comments on the research presented in this dissertation. Also, I would like to thank members of our research group, Indranil Roychoudhury, Aparna Barve, Ashraf Tantawy, Chetan Kulkarni, and Joseph Hall, who my interactions with have also improved the quality of this work. In addition, thanks to Nagabhushan Mahadevan, for his help in the implementation of the diagnosis approach within the FACT architecture.

The work on the Advanced Diagnostics and Prognostics Testbed (ADAPT) at NASA Ames Research Center has formed a large part of this dissertation. Therefore, I owe thanks to members of the ADAPT team, Ann Patterson-Hine, Scott Poll, Adam Sweet, Joe Camisa, David Nishikawa, David Hall, Serge Yentus, Charles Lee, Christian Neukom, John Ossenfort, and David Garcia, who helped in basic testbed operation, model-building, code development, and participated in research discussions.

This work was supported in part by the National Science Foundation under Grants CNS-0452067 and CNS-0615214, by the National Aeronautics and Space Administration under Grants NSF-NASA USRA 08020-013 and NASA NRA NNX07AD12A, and by Vanderbilt University under a University Graduate Fellowship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

For complex engineering systems, online fault diagnosis is key to achieving safe and reliable operation in spite of system faults. Faults must be detected and isolated early, so that reconfiguration actions can be taken and catastrophic situations can be avoided. Many complex systems are embedded, that is, they consist of a physical plant with a controller that is implemented in software. The system may undergo configuration changes through the actions of the controller. Such systems can be modeled best as *hybrid systems*, where the dynamic behavior evolves continuously in time, but may undergo discrete changes as the system configuration, or *mode*, changes. Due to the complexity of such modern engineering systems, formal methods are required for systematic design, analysis, and implementation of hybrid system diagnosers. Model-based diagnosis provides a formal framework to achieve these objectives [1].

In model-based diagnosis, fault diagnosis algorithms reason about differences in observed behavior from that predicted by a model, by exploiting the analytical redundancy between the system and its model. Logic-based approaches provide a formal logic framework for defining and solving the diagnosis problem, however, they typically deal with only static systems [2–4]. Discrete-event system approaches abstract time to event sequences, but largely ignore the continuous dynamics of the system [5–8]. Continuous systems approaches represent the evolution of system behavior as a function of time, but cannot deal with changes in system configuration [9–13]. Therefore, these traditional approaches do not apply to hybrid systems, where diagnosis must deal with the evolution of both the continuous and the discrete aspects of system behavior [14–19]. The previous approaches, however, provide insights that can be leveraged to provide better solutions to hybrid systems diagnosis.

This dissertation develops an event-based framework for fault diagnosis in hybrid systems that extends the TRANSCEND and Hybrid TRANSCEND methodologies [14, 20–22]. In these frameworks, diagnosis is based on analysis of fault transients, where observed deviations from nominal behavior expressed in qualitative form are compared against qualitative predictions of faulty behavior, i.e., *fault signatures*, to isolate faults. We extend the previous work in four significant ways. First, we extend the discriminatory ability of the diagnosis algorithms by including *relative measurement orderings*, which define predicted temporal orders of measurement deviations due to faults. Second,

we extend fault isolation procedures that are based on the single fault hypothesis to multiple fault diagnosis. Third, we extend the previous fault isolation from only *parametric fault* isolation algorithms to include *discrete faults*, which are defined by unanticipated changes in the system mode. Fourth, we merge all these contributions into a novel event-based framework for hybrid systems diagnosis. A formal methodology is developed to design event-based diagnosers, and establish and analyze notions of diagnosability for a system. Further, we perform experimental studies on two practical systems, (i) a formation of mobile robots, and (ii) a complex electrical distribution system.

In the rest of the chapter, we first describe the motivation for our work, followed by the specific research objectives. We then describe the specific contributions of the dissertation, and conclude with the organization.

Motivation

Modern engineering systems in application domains such as robotics, electronics, and aeronautics are typically made up of a large number of interacting components. These systems and components are hybrid in nature, that is, they exhibit mixed continuous and discrete behavior. Model-based diagnosis for such systems is a challenging task. The involved models must accurately define the dynamic system behavior in each mode of operation. Practical systems are typically highly reconfigurable, therefore, they operate in a large number of modes. Further, transitions between modes may be controlled, which are based on known switching inputs, or autonomous, which are based on internal system variables. The large number of possible system modes and the inability to predict mode transitions after fault occur complicates the diagnosis task, and reduces the discriminatory ability of diagnosis algorithms.

Model-based diagnosis takes advantage of the analytical redundancy between expected behavior defined by a system model, and observed behavior provided by the sensors. Although modeling is a challenging task, especially for hybrid systems, if an appropriate model can be developed, then a model-based approach provides a better solution to the diagnosis problem, because the knowledge contained in the model is leveraged for more precise and accurate diagnosis. Further, model-based solutions can provide analysis tools to verify important properties of the model, such as diagnosability.

In model-based diagnosis, one must take advantage of as much of the diagnostic information that can be extracted from the model. The only form of information employed in typical approaches for continuous and hybrid systems is that of fault signatures, defined by the deviations from expected,

nominal behavior [9, 10, 14, 20]. Discriminatory ability can be improved by adding more sensors, however, in many applications, adding extra sensors may not be cost-effective. Moreover, the addition of certain sensors may not be physically possible. Sensors may also be faulty, which adds to the complexity of diagnosis. Uncertainty in the signatures or in the mode of the system decreases the discriminatory ability of the algorithms. The full discriminatory power of a set of measurements must be exploited so that faults can be diagnosed with the smallest number of measurements necessary. Approaches in discrete-event systems use temporal information in the form of events, and recent work in continuous systems diagnosis has shown that similar information can be extracted from continuous models to enhance diagnosis [23–28]. This type of information can be employed in addition to traditional fault signatures to enhance fault isolation.

Hybrid systems may undergo many different types of faults, so a comprehensive diagnosis framework geared toward real-world applications must handle different fault types. In hybrid systems, faults may not always be characterized by changes in system parameters, i.e., parametric faults, but also attributed to undesired changes in the configuration of the system, i.e., discrete faults. Most current hybrid systems diagnosis methods cater exclusively to either parametric faults [14, 17, 29] or discrete faults [15, 30–34], but few to both [19, 35–37]. In order to be robust, hybrid diagnosis strategies must deal with both fault types.

Fault isolation is also complicated by the occurrence of multiple faults. In general, a number of different faults can occur in complex systems, and the likelihood of multiple faults occurring increases in harsh operating environments. Fault diagnosis schemes that do not take into account multiple faults run the risk of generating incorrect diagnoses or even failing to find a diagnosis after faults occur. Multiple fault diagnosis is further complicated by the fact that multiple faults' effects may mask one another, thus making it difficult to differentiate between multiple fault candidates [38, 39]. Due to fault masking, multiple faults can affect the measurements in many different ways, and this adds uncertainty, which in turn reduces the discriminatory ability of diagnosis algorithms. Further, the reasoning performed over the space of multiple fault hypotheses is expensive, so diagnosis algorithms should be designed as efficiently as possible.

With all of these issues, diagnosis in hybrid systems becomes a challenging task. In order to facilitate diagnostic analysis of hybrid systems, formal methods are required to establish and verify important properties of systems, such as diagnosability. Formal methods also allow systematic design, analysis, and implementation of diagnosers. Since diagnosis in hybrid systems is event-based, i.e., events that change the mode of the system and events signifying fault occurrence, an event-based framework to support such formal methods offers significant advantages.

While developing the necessary theory is important, application to real systems is the goal. Diagnosis algorithms should be tested and evaluated on real systems with practical significance. In these systems, factors such as sensor noise and model inaccuracy make diagnosis challenging. Therefore, there is a need to evaluate the robustness of the diagnosis algorithms for different fault and noise magnitudes. Accurate simulation models of the system are required for this purpose. Further, it is important to execute the diagnosis algorithms on the physical systems, where model uncertainty is always present, and complicates the diagnosis task. Experiments on real systems often brings to light new challenges that need to be addressed for actual deployment of the diagnosis algorithms.

Contributions

This dissertation develops an event-based framework for fault diagnosis in hybrid systems that extends the TRANSCEND and Hybrid TRANSCEND methodologies [14, 20–22]. In these frameworks, diagnosis is based on analysis of fault transients, where observed deviations from nominal behavior are matched against qualitative predictions of faulty behavior, known as fault signatures, to isolate faults. Significant work has been done within TRANSCEND that addresses the issues of modeling, system monitoring, fault detection, and fault isolation for hybrid systems. However, the framework does not utilize the temporal orders of measurement deviations within fault isolation and considers only single and parametric faults. This work extends the previous fault isolation algorithms to incorporate temporal orders of measurement deviations, address multiple faults, and provide an integrated framework for diagnosis of both parametric and discrete faults in a formal event-based framework. We also make necessary extensions to the other parts of the framework; specifically, we extend the modeling language and symbol generation for discrete faults.

Our specific contributions are as follows.

1. **Relative Measurement Orderings.** We introduce the notion of *relative measurement orderings*, which define predicted temporal orders of measurement deviations due to faults. In conjunction with fault signatures, measurement orderings allow us to define event-based *fault models*, where the events are measurement deviations. Relative measurement orderings increase the discriminatory ability of the diagnosis algorithms, which reduces ambiguities in the diagnosis results, and generates the fault isolation results faster, so corrective actions can be taken sooner. We establish how relative measurement orderings can be extracted from the sys-

tem model, and correspondingly enhance the prediction algorithm of TRANSCEND to predict relative measurement orderings as well as fault signatures.

2. **Discrete Faults.** We extend the previous fault isolation schemes designed to isolate *parametric faults*, defined by unanticipated changes in a system parameter value, to *discrete faults*, which are defined by unanticipated changes in the mode of a switching component. We extend the modeling language and accompanying diagnosis model to incorporate discrete faults. We extend the hypothesis generation algorithms of TRANSCEND to identify discrete faults as possible causes of faulty behavior, and the prediction algorithm to predict the effects of discrete faults. Further, we augment fault signatures with an additional symbol that is useful in distinguishing between parametric and discrete faults, and extend the symbol generation procedure accordingly.

3. **Multiple Faults.** We extend the fault isolation scheme to perform multiple fault diagnosis. We illustrate how the effects of multiple faults can be formed in a consistent manner from the individual faults, for which relative measurement orderings are necessary. We improve the efficiency of multiple fault isolation by extending notions of *minimality* used in logic-based diagnosis, and limiting the cardinality of the multiple fault hypotheses.

4. **Event-based Diagnosability Analysis and Diagnoser Design.** We construct our approach as a formal, event-based framework for hybrid systems diagnosis. Under this framework, we can create event-based diagnosers similar to those used in discrete-event systems diagnosis, where events are defined as measurement deviations and controlled mode changes. Using this framework, we establish notions of diagnosability for a system, and show how the diagnosers can be used to verify diagnosability of a system. Moreover, we offer a spectrum of implementations of the event-based diagnosers that trade off space- and time-efficiency.

5. **Experimental Case Study for Robot Formations.** We provide a case study to illustrate our diagnosis approach using formations of mobile robots. Because the system is inherently distributed, we apply the distributed TRANSCEND approach [40, 41], but extend it to incorporate relative measurement orderings. We present diagnosis models for robot formations, and provide a diagnosability analysis which shows that relative measurement orderings are required to discriminate between faults on different robots. We apply the distributed diagnosis approach experimentally on a formation of four mobile robots.

6. **Experimental Case Study for Electrical Power Distribution Systems** The mobile robot system only requires continuous systems modeling and diagnosis. We also apply our approach to a realistic electrical distribution system of a spacecraft, the Advanced Diagnostics and Prognostics Testbed (ADAPT) at NASA Ames Research Center [42,43]. ADAPT contains over 100 sensors, has over 170 possible faults, and incorporates over 50 circuit breakers and relays that configure the system into different modes. The system also contains lead-acid batteries that have highly nonlinear behaviors. Because of these elements, ADAPT serves as a challenging testbed for hybrid systems diagnosis. We consider a subset of ADAPT that includes a battery connected to two DC loads through relays. We provide diagnosability analysis of the selected subsystem, and extensive simulation results that investigate the effect of fault magnitude and sensor noise on the precision and accuracy of the diagnosis results. Further, we provide experimental results obtained from running the diagnoser on the actual system to demonstrate the approach.

Organization

Chapter II presents an overview of model-based diagnosis. First, we examine the problems that need to be solved for diagnosis of hybrid systems. Then, we examine the logic-based approaches to model-based diagnosis, which provide a formal foundation to the problem. Logic-based approaches typically deal only with static systems, so we then describe approaches in discrete-event systems, where time is abstracted to sequences of events. Because time is abstracted to events, it becomes difficult to achieve fast fault detection and isolation. So, we then move to continuous systems diagnosis, which represent the dynamic behavior of the system as a function of time. Continuous systems representations do not account for changes in system configuration, therefore, we examine model-based diagnosis methods for hybrid systems. Each modeling framework provides insights into how to best use certain types of information, and we leverage this information throughout the dissertation.

Chapter III formulates the problem of fault diagnosis in hybrid systems. Diagnosis becomes difficult in hybrid systems when mode changes, multiple fault types, and simultaneous faults are considered. We outline the specific challenges that arise, and overview the assumptions we make to enable the problem to be more manageable for many practical systems. We then outline our general architecture for diagnosis, and summarize the features of TRANSCEND (e.g., [14, 20–22, 44–51]) that we use.

Chapter IV describes our modeling framework. We adopt hybrid bond graphs due to their many advantages for model-based diagnosis [21]. We extend the modeling language to account for discrete faults and to incorporate sensor models. We then present our diagnosis model, the temporal causal graph [20]. We extend the temporal causal graph to explicitly account for the causal relationships between discrete faults and the system variables, and also to account for sensor models.

Chapter V presents the qualitative fault isolation framework for continuous systems diagnosis. We extend the TRANSCEND methodology by incorporating relative measurement orderings and including algorithms for isolation of multiple faults. We describe how fault hypotheses are generated, and how predictions of faulty behavior are computed. We extend prediction to include relative measurement orderings, and develop event-based fault models that combine fault signatures and measurement orderings. We then describe the online fault isolation algorithms that use the event-based fault models.

Chapter VI presents the qualitative fault isolation framework for hybrid systems diagnosis. We extend the Hybrid TRANSCEND methodology to account for both parametric and discrete faults, and include algorithms for isolation of multiple faults. We describe how fault hypotheses are generated, and how predictions of faulty behavior are computed as extensions of our updated continuous systems approach. We then describe the online fault isolation algorithms that use the event-based fault models for hybrid systems.

Chapter VII presents systematic design procedures that produce event-based diagnosers, by compiling the event-based fault models and the fault isolation algorithms into a computational structure that can be used for both efficient online diagnosis, and for examining diagnosability properties of the system. We first develop notions of diagnosability within our framework, and then describe our representation of event-based diagnosers. We then describe the design procedures for single- and multiple-fault diagnosis in continuous systems, and diagnosers for hybrid systems, which account for faults as well as controlled and autonomous mode transitions. We then provide a spectrum of implementations of the event-based diagnosers that trade off space- and time-efficiency.

Chapter VIII provides a case study for our approach on formations of mobile robots, demonstrating diagnosis of single faults in continuous systems. Because robot formations are distributed systems, we present a distributed architecture of our approach that extends distributed TRANSCEND [40, 41] to incorporate the use of relative measurement orderings, which results in distributed event-based diagnosers for each robot that do not require a centralized diagnosis coordinator. We present a general modeling and control framework for robot formations, and describe the distributed diagnosis problem formulation and architecture. We then provide detailed diagnosabiltity and di-

agnoser design results for the formation systems, and experimental results demonstrating the effectiveness of the approach.

Chapter IX provides a second case study that focuses on hybrid systems. We apply our algorithms to the Advanced Diagnostics and Prognostics Testbed (ADAPT) at NASA Ames Research Center [42, 43]. ADAPT brings the challenges of hybrid systems diagnosis to the forefront. The system has a number of relays and circuit breakers, which makes the system behavior naturally hybrid. Further, it contains lead-acid batteries, the behavior of which is highly nonlinear. Further, both parametric and discrete faults can occur, and, therefore, ADAPT serves as a promising testbed to validate our overall approach. We adopt a subset of ADAPT to demonstrate our approach, and first present the component models. We then provide diagnosability results for the selected subsystem, as well as detailed simulation experiments that vary fault magnitude and sensor noise to examine the robustness of our algorithms. We also provide experimental results that were obtained by running our diagnoser on the actual testbed.

Chapter X summarizes the contributions of this dissertation. We then point out the current limitations of the approach, and describe directions in which this work may be improved in the future.

CHAPTER II

RELATED WORK IN MODEL-BASED DIAGNOSIS

Model-based diagnosis methods reason over differences between expected behavior defined by a system model, and observed behavior provided by the sensors. Modeling is a challenging task, and the quality of the model impacts the quality of the diagnosis results. But, if an appropriate model can be developed, then a model-based approach provides a better solution to the diagnosis problem, because the knowledge contained in the model is leveraged for more precise and accurate diagnosis [2–6, 10, 14, 18–20, 28, 39, 52–55]. Further, model-based solutions can provide analytical methods to verify important properties, such as diagnosability.

Approaches to model-based diagnosis mainly differ by the type of modeling representation used. The manner in which the system is modeled largely influences the fault diagnosis techniques, and the diagnostic precision that they can obtain. *Logic-based* approaches represent the system using a formal language such as propositional or first-order logic [1–3, 56–61]. They typically abstract time out of the system behavior description, thus they cannot be directly used to building diagnosers for dynamic systems. *Discrete-event system* approaches abstract time by modeling system behavior as a sequence of events, and the corresponding diagnosis schemes are based on analyzing observed event sequences [5–8, 62–69]. *Continuous system* approaches employ a continuous- or discrete-time representation of the system, and use quantitative and/or qualitative methods for diagnosis [9–13, 23–26, 52, 70–74]. *Hybrid system* approaches combine continuous and event-based behaviors, and are the most general, but also the most complex [15, 17–19, 29–35, 75–77]. Hybrid diagnosis methods range from estimation-based approaches to mixed qualitative and quantitative approaches.

This dissertation focuses on diagnosis of hybrid systems. Hybrid systems combine continuous dynamic behavior and discrete mode behavior, where the continuous dynamics change from mode to mode. Hybrid systems cover a wide class of systems and system behaviors, which implies that model-based diagnosis of hybrid systems can be applied to a broad class of applications. For example, hybrid systems are very important for modeling embedded systems, because they consist of physical plants with continuous behaviors combined with computer processes that drive the plant to particular operating modes through switching actions.

Diagnosis of hybrid systems is an open and active research area. As described in Fig. 1, model-based diagnosis of hybrid systems can be decomposed into four problems [14, 37]:

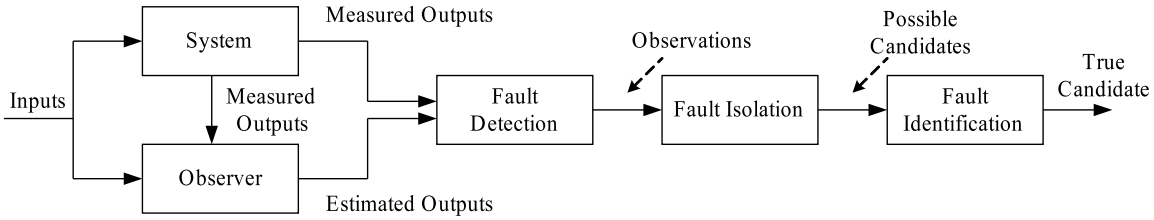


Figure 1: Computational architecture for model-based diagnosis of hybrid systems.

1. **Estimation.** In dynamic systems, the behavior evolves over time, therefore, continuous monitoring is necessary in order to predict the time-varying behavior. Some techniques utilize a state-space or input-output model for designing an observer, which computes expected behavior given the system model, the inputs, and the measured outputs, and can be robust to sensor noise, model uncertainty, and disturbances. Observers must also be efficient enough to allow for real-time tracking, but in hybrid systems, tracking becomes complicated because the state consists of both a continuous component and a discrete component. Hybrid observers for diagnosis have been implemented using switched Kalman filters, banks of Kalman filters, and particle filters [16, 30, 32–34, 50, 75].

2. **Fault Detection.** Given both measured and estimated outputs, observed differences are computed, and a decision must be made as to whether the differences implicate the presence of a fault in the system. Fault detection takes as input the difference between expected and actual outputs, defined as the residual, and outputs a true or false value indicating fault presence. In dynamic systems, sensor noise, model uncertainty, and disturbances make fault detection challenging. A robust fault detection module is necessary to perform the decision test. For example, one type of fault detector takes the form of a special observer known as a residual generator. Robust residual generators that respond only to subsets of faults can also be designed [9–12].

3. **Fault Isolation.** Given that a fault has occurred, fault isolation reasons about the differences between model-predicted and observed behavior, and establishes possible candidates as combinations of faults that can explain the observed behavior. In single fault diagnosis, the goal is to obtain a unique single fault that can explain the observations. It may not always be possible to determine a unique candidate given the type of information used for diagnosis and the sensors available on the system. In multiple fault diagnosis, the goal is expanded to obtain sets of faults that, taken together, explain the observations. However, due to the

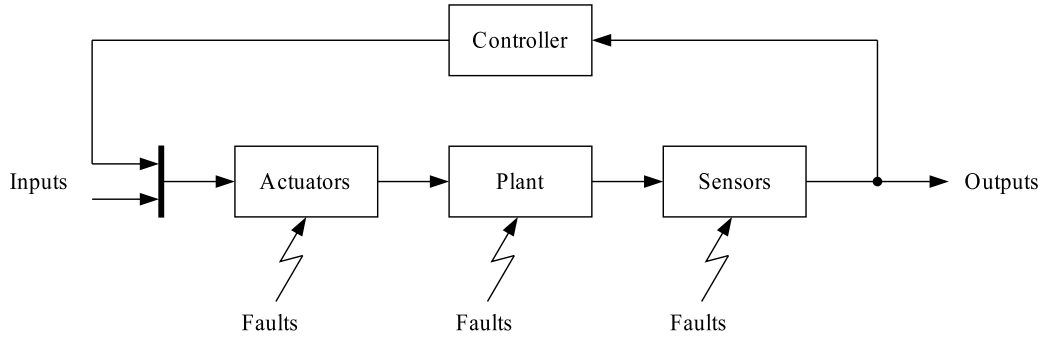


Figure 2: Faults in dynamic systems.

interactions of multiple faults, which may cause fault masking, it is often hard to obtain a unique candidate [38, 39].

4. **Fault Identification.** If fault isolation discovers a unique fault set, then fault identification computes the magnitudes of the faults that match the observations most closely. If multiple fault sets remain after isolation, then this must be performed for each fault set, and the fault set that obtains the closest match is determined to be the true fault set [14, 22, 45]. Some approaches combine fault isolation and fault identification [15, 17, 32, 78, 79].

The objective of diagnosis is to establish which possible faults or combinations of faults match the observed system behavior. As shown in Fig. 2, faults may manifest in different parts of the system, namely, the actuators, the plant, and the sensors. In hybrid systems, faults can occur as parameter value changes in continuous modes of operation, and are termed *parametric faults*. Faults can also occur in the form of mode-changing behavior, which are represented as unexpected changes in system mode and are termed *discrete faults*. Logic-based and discrete-event system approaches deal with faults that are characterized as discrete, while continuous systems approaches deal with faults that are characterized as parametric. Hybrid systems approaches must deal with both types of faults.

Because of the different types of faults and the different modeling representations used, fault diagnosis can be performed in many different ways. In order to provide a context for the research described in this dissertation, we provide a taxonomy of approaches in model-based diagnosis. First, we describe logic-based approaches, then discrete-event approaches, continuous systems approaches, and finally hybrid systems approaches. We conclude with a summary of the different approaches and describe how the insights they provide connect to the results of this dissertation.

Logic-Based Diagnosis

Performing model-based diagnosis requires modeling the system of interest, and reasoning about differences between model-predicted and observed behavior. The work in discrete, logic-based approaches provides a formal foundation for model-based diagnosis [1].

In logic-based approaches, the system is modeled in a formal logic framework. The system and relevant domain knowledge are captured in a system description, SD. The set of components, $COMPS$, is assumed to be a set of logical constants. Faults are simply modeled as incorrect behavior of a component. If a component $c \in COMPS$ is not working correctly, then the statement $ab(c)$ is true, and implies that the component is abnormal. The set of observations, OBS, is, like SD, a set of logical sentences, and describes the current system state as observed by the sensors. The goal of diagnosis is to find all $c \in COMPS$ for which $ab(c)$ is consistent based on OBS and SD.

There are a number of challenges in applying logic-based approaches to complex dynamic systems. Unless the logic is restricted, diagnosis in this paradigm can be computationally inefficient because it is based on theorem-proving, truth maintenance, or manipulation of logical expressions, which are all NP-hard in general [80]. Modeling complex dynamic systems in a suitable logic is also very difficult and tedious, and puts a lot of burden on the modeler. How to represent time at the appropriate level of abstraction also becomes a difficult issue to resolve. A concise representation of the system with only the information needed for diagnosis is necessary in order to keep these approaches as efficient as possible.

Consistency-based Diagnosis

In *consistency-based* diagnosis, a set of faulty components is assumed, and checked for consistency with the system model and the observations. A consistency-based diagnosis theory, using the SD, $COMPS$, OBS, and $ab(\cdot)$ formulation, and an algorithm to compute diagnoses based on logical consistency, is described in [2]. The advantage of the approach is that, given an adequate system description in a suitable logic (e.g., first-order, temporal, dynamic), the approach can be readily applied.

The General Diagnostic Engine (GDE) implements a similar approach that is developed to specifically handle multiple fault diagnosis efficiently [3]. To help reduce the exponential nature of the problem, fault candidates are handled using minimal sets that can explain all of the observations. Using minimal sets is reasonable because, in most cases, if some set of failed components $C \subseteq COMPS$ is consistent with the system observations, any superset of C is also consistent. Any component may

be faulty, but its faulty behavior may not have been observed yet. Therefore, finding the minimal sets is sufficient to characterize all possible diagnoses of the system. The approach is also incremental, so when a new observation is made, a *conflict set* is generated. A conflict set represents the set of assumptions that supports a symptom. The current set of fault candidates is then updated to account for the new conflict. The key advantage of the approach is the method of manipulating multiple fault candidates efficiently, which may be extended to dynamic systems [38, 39].

In GDE, components can be in one of two states, correct or faulty. In many practical situations, however, it is not enough to know that a component is faulty, but also in what way it has failed. The concept of behavioral modes is developed in [56], where the normal behavior of components is described as well as specific faulty behaviors, for each possible known failure mode. Typically, the behavioral specification also includes a mode with no constraints, in order to model unknown failure modes. Diagnosis identifies which components have failed and in what modes, and reduces to finding consistent assignments of modes to components.

Temporal Diagnosis

While logic-based approaches often abstract time out of the models, temporal diagnosis methods retain time in a temporal logic framework. A diagnosis method performed by finding possible sequences of actions or events that brought the system to the current state is presented in [59]. Possible trajectories leading to the current state are found by applying theorem proving methods. The trajectories may contain faults that can explain the current history of observations, thus providing the diagnosis results. In this approach, the system domain is axiomatized in the situation calculus, where time is abstracted using events or actions.

A related approach that develops a temporal diagnosis framework is described in [58]. Temporal diagnoses are defined as sequences of events that have occurred in the system. Systems are modeled in a discrete-event framework using qualitative methods, and the goal is to find possible event sequences that are consistent with the sequence of observations.

Other work has focused more specifically on temporal logics [60, 61, 81]. A temporal constraint language that includes constraints to define different qualitative and quantitative temporal relations between actions is provided in [60]. An efficient algorithm for abductive diagnosis using this language is presented in [61]. As with the previous logic-based approaches, it is often difficult to model complex, dynamic systems using a formal temporal logic language.

Discussion

Logic-based approaches provide a formal reasoning framework for model-based diagnosis using theorem proving. For complex systems, however, especially hybrid systems, modeling in a formal logic framework is cumbersome, and leads to large models where establishing results using theorem proving becomes very inefficient, especially when using temporal logics. Although most consistency-based logic approaches deal with static systems, some of the concepts can be applied in diagnosis of dynamic systems. For example, the concept of minimal diagnoses can be applied to diagnosis of faults in continuous systems if conflicts can be generated correctly with respect to time [39, 82]. We use an equivalent form of conflicts and show how they can be computed correctly in Chapter V.

Discrete-event Systems Diagnosis

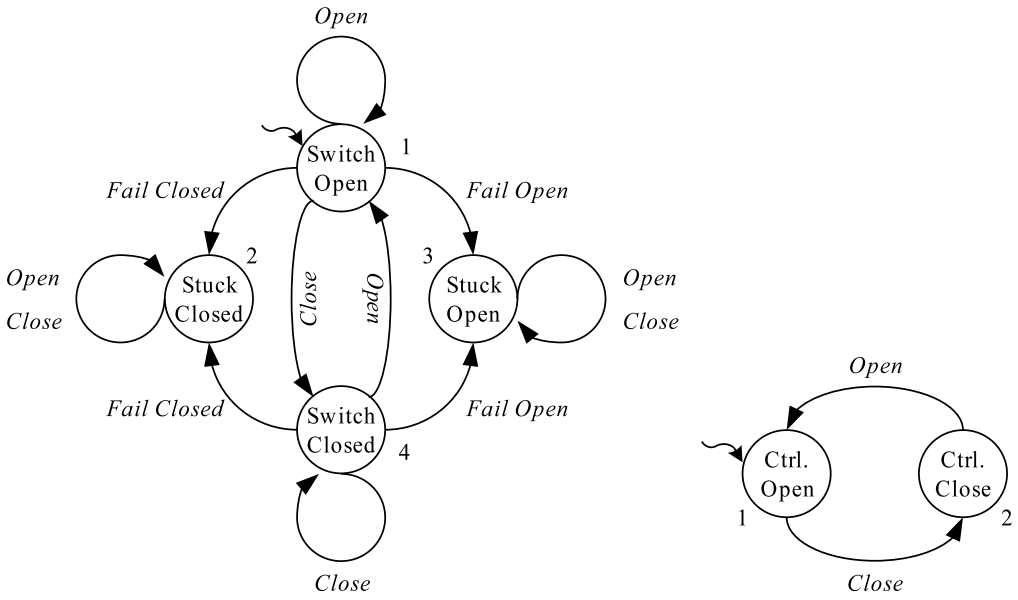
Often, temporal information can be encoded as sequences of events, which may correspond to discrete observations, controller actions, alarms from system components, and other relevant system events. *Discrete-event systems* (DES) abstract time to discrete points at which events occur. A DES model is represented by a set of states, a set of observable and unobservable events, and transitions between states that depend on the occurrence of events [83].

DES diagnosis methods typically compile a diagnoser for the system based on its DES model [5, 6]. The compiled diagnoser avoids the need for theorem proving, thus allowing for more efficient diagnosis. In DES, faults are modeled as unobservable events. The diagnoser tracks observed event sequences links them back to the unobservable fault events.

Example. Consider the models of a simple switch and its controller given in Figs. 3a and 3b, respectively. The complete system model is given as a synchronization of these two component models based on the shared *Open* and *Close* events. The unobservable fault events are *Fail Open* and *Fail Closed*, in which the switch becomes stuck in its current state and fails to respond to further controller commands. It is assumed that a sensor exists that determines whether the switch appears to be on (1) or off (0).

DES diagnosis is based on relating observed system events to the system's state. Ideally, fault occurrences will eventually generate a sequence of observable events that are different from nominal and other fault sequences. For example, a stuck-closed failure in the switch can be diagnosed when an *Open* command is given but the sensor still reads 1.

There are many significant challenges to applying DES diagnosis methods to complex systems. Describing a system in a discrete-event framework is the foremost issue. To apply DES methods, the



(a) Switch. (b) Controller.

Figure 3: DES models of a switch and its controller.

system behavior must be abstracted to event sequences that include sufficient detail so that faults can be detected and isolated. When timing information is considered, the complexity of the model grows. Detailed models are needed to improve diagnostic precision, but they become cumbersome to develop, and they often degrade the efficiency of the diagnosis algorithms. More abstract models offer greater algorithmic efficiency but less diagnostic precision. DES methods can be untimed, in which the only timing information is ordering of events, or timed, in which timing information such as time bounds on event occurrences are included.

Untimed Discrete-event Systems Diagnosis

An event-based modeling and diagnosis framework for systems in the DES framework is described in [5, 62]. Individual system components are modeled as DES, then composed to form the complete system model. This model is used to construct a diagnoser that provides estimates of the system state under nominal and faulty conditions. Unique fault isolation is formalized using the notion of *diagnosability* applied to the formal language represented by the DES with respect to observable events. Essentially, a DES language is diagnosable if, after the occurrence of a fault, there exists a finite sequence of observable events that uniquely identifies that fault.

The diagnoser, constructed off-line from the model, serves as an extended observer. Each diagnoser state has estimates of the current system state and failure labels. The diagnoser for the switch

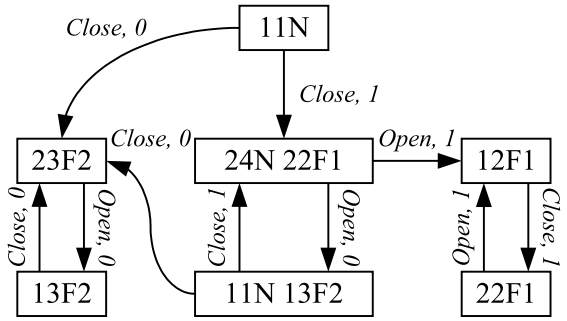


Figure 4: DES diagnoser the switch and controller system.

and controller system is shown in Fig. 4. The first number in the state refers to the state of the controller, and the second number refers to the state of the switch. Failure label N represents normal, F1 represents the occurrence of *Fail Closed*, and F2 represents the occurrence of *Fail Open*. Each state-label pair corresponds to a possible system state given the observations. In the diagnoser events, the first event represents the observable controller event, and the second the sensor reading. A failure label associated with a state implies that a failure of that type must have occurred. When an event is observed, the diagnoser provides the new state estimate based on the observed event and the previous state. If a fault occurs in the system, and the system is diagnosable, then the diagnoser will eventually reach a state that corresponds to that failure type. For example, if the controller tries to open the switch but the sensor reads 1, the diagnoser's state estimate is that the switch is in the stuck closed state and that *Fail Closed* has occurred. This corresponds to the diagnoser state labeled by 12F1.

A state-based DES diagnosis approach is described in [6]. Being state-based, the focus is on determining system condition (failure mode), rather than the explicit failure events that have occurred. This is because, in most practical settings, system models are constructed by composing several smaller component models, each usually having a single nominal mode and a few failure modes. Thus, there is a direct relation between system state and failure mode. The approach assumes that the state set of the system can be partitioned according to the system condition, defined by a single nominal mode and several failure modes of the system. In order to perform state-based diagnosis of DES, the system model is augmented to include a set of outputs and an output map. System condition is then inferred from the output sequence. A diagnoser that determines system condition is constructed in [6]. Diagnoser states contain an output, possible system states consistent with the output sequence, and possible system conditions associated with these states. The diagnoser, its construction, and the diagnosability conditions are analogous to those in the event-based approach.

Diagnosis of DES based on linear-time temporal logic (LTL) specifications is developed in [63], where it is argued that it is more natural to express failures in LTL than in a formal language. The DES model of the system is assumed to be given, and is extended by attaching LTL predicates to each state. The specifications are formulated in terms of these predicates. With each fault there exists an associated LTL specification. Violation of this specification indicates the occurrence of the fault event. This approach is more general than the previous approaches, because it can specify and diagnose properties such as safety, liveness, invariance, and stability [63]. A system is said to be diagnosable if for each specification, the occurrence of an observable event sequence that can be linked back to a violation of the specification can be detected after finite delay. This is a generalization of the diagnosability definition for the event-based approach. An algorithm for diagnoser construction based on the given specifications is provided in [63]. For each specification, a diagnoser is constructed as an automaton accepting only nonfaulty traces with respect to the specification. The occurrence of an observable event trace not accepted by the diagnoser indicates a specification violation.

Specifying the behavior of a complex system using the DES modeling paradigm has its advantages and disadvantages. One clear advantage is the simplicity of the diagnosis approach. The diagnoser simply tracks the observable event sequences in the system, and entering certain diagnoser states indicates which failures have occurred. Generation of the diagnoser or set of diagnosers is also an automatic process given the system model. For diagnosis of complex systems, however, a DES model may specify behaviors at too high a level of abstraction, and information which is crucial to diagnosing critical failures may be lost. As the abstraction level is decreased, the size of the model becomes too large and difficult to develop.

Timed Discrete-event Systems Diagnosis

In many cases, faults do not change system event sequences, but only change timing characteristics of the system [7, 8]. The timing of an event sequence can change, or events can stop occurring. Timed DES approaches address these issues. In developing a timed model, usually two machines are constructed, an activity transition graph (ATG), and a timed transition graph (TTG) [84]. The ATG is at a higher level of abstraction and includes time bounds on when events should occur. The TTG is derived from the ATG, and introduces an additional observable event that represents the tick of a global clock. Timing characteristics of the system are built into the system model using the clock tick event.

An extension of the untimed event-based DES diagnosis approach to timed DES is given in [7]. In the ATG, event labels are associated with timers and transitions are associated with lower and upper time bounds. The lower time bound describes the delay that must occur before the transition is enabled, and the upper bound describes the maximum delay that may occur before the transition is taken. Based on the ATG, the TTG is generated, which incorporates the timing characteristics specified by the timers and transition time bounds using the observable global tick event. In this framework, a timed DES is said to be diagnosable if its TTG is diagnosable under the event-based diagnosability definition presented in [62]. Therefore, the results of the untimed event-based approach directly apply and the diagnoser is generated in the same manner using the untimed model, but using the new observable tick event.

The untimed state-based approach can also be extended in a similar manner to timed DES if the global tick is treated an another system observation. However, this can create very large diagnosers and increase the online computation requirements [8]. An alternative approach that does not update the state estimate at clock ticks but instead incorporates predictions of the system condition is given in [8]. Clock events are projected out of the system model and a timed finite state machine is constructed which associates timed delays with each transition. This increases design time complexity, but reduces on-line computations and diagnoser size. The diagnoser structure is the same as in the untimed state-based approach, but is augmented to included predictions of the system state. In this way, permanent failures, which result in no new output symbols being generated, can be diagnosed.

Timed discrete event behavior can also be modeled using timed automata models. A diagnosis method for timed automata that extends the untimed event-based approach is described in [64]. Based on the system model, which includes fault events, a diagnoser is constructed to isolate the occurrence of unobservable fault events.

Alternatively, chronicles have been used to model timed event traces in systems, and have been applied mostly to telecommunication systems [65–67]. Chronicles are patterns of event traces that include temporal constraints and represent the possible evolutions of the system behaviors for each fault. Therefore, chronicles represent direct symptom to fault knowledge, thus they are very efficient for online diagnosis [65,66]. As events occur in the system, they are matched against known chronicles to determine which faults are present. However, a major drawback is that generating chronicles can be difficult. Learning approaches for generating chronicles are described in [66]. Learning methods, since they are based on training, require a lot of data to generate correct and reliable models of

system behavior. In many systems, faulty data for training may be undesirable or impossible to obtain.

A significant issue with timed DES approaches is the use of time bounds. Temporal constraints in timed automata, chronicles, and other timed DES models that involve bounded intervals cannot be used for continuous systems except in trivial cases, because the intervals are functions of the fault magnitude. For continuous systems, time bounds depend on the system dynamics, which depend on the time constants of the system, the system inputs, and fault magnitudes. Therefore, in many practical cases, time bounds cannot be acquired. Quantization of the continuous dynamics is required to address this issue.

A quantization scheme is used to model continuous systems with discrete inputs, sensors, and faults based on a timed discrete event model in [68, 69]. The quantization allows automatic derivation of a DES model based on a continuous-time model. Starting with a state-space formulation of the system, the state space is partitioned and events are defined for crossings of those partitions. The quantizer generates the timed event sequence when events occur in the system. The diagnoser then uses these timed event sequences, along with timed input sequences, to diagnose the system. Quantization, however, inherently introduces nondeterminism in the model [68, 85]. From a known quantized state and known input, it may be possible for the system to enter more than one new quantized state, since the exact point in the quantized space is unknown. Because of the nondeterminism, the diagnostic system must track multiple possibilities simultaneously. Having a nondeterministic model degrades the performance and increases the computational requirements of diagnosis algorithms.

The quantization approach seems appropriate only for systems with discrete inputs, discrete sensor measurements, and discrete faults, because the inputs, measurements, and faults all have to be discretized. If discrete sensors are available, then the approach is practical. If the sensors are not discrete, then the discretization causes a loss of information. This may make fault detection less sensitive, and important diagnostic information could be lost because of the measurement abstraction. The finer-grained the quantization and the greater the range of possible inputs, the larger and more complex the model will be. Faults also have to be quantized by their magnitude. If faults can occur in different system states (as is usually the case), then the DES model becomes very large, just as with the other timed DES approaches. This methodology cannot handle multiple faults in a feasible manner, because the state space increases exponentially with the number of possible faults.

Discussion

DES models abstract away details of the continuous dynamics that are often important for fast and precise fault detection and isolation. Timed models can capture some of the dynamics beyond event ordering, but require quantizations, which are expensive, introduce nondeterminism, and reduce sensitivity to fault detection. Further, DES approaches lack sophisticated tools to deal with system noise, model uncertainty, and disturbances, which are important factors in dynamic system diagnosis. In hybrid systems, models for diagnosis need to seamlessly mix both discrete and continuous behaviors of the system. DES approaches do, however, provide a well-developed framework for event-based diagnosis, and these techniques can be leveraged in building continuous and hybrid system diagnosers [82, 86]. In Chapters V, we will show how event-based models of faulty behavior in dynamic systems can be developed from continuous system models, and in Chapter VII, we will show how this information can be used to compile event-based diagnosers for continuous and hybrid systems.

Continuous Systems Diagnosis

In continuous systems diagnosis, the system is modeled using differential or difference equations. In qualitative methods, the system equations are abstracted to a set of qualitative differential equations [73, 74, 87]. Quantitative methods focus on robust solutions for the estimation and fault detection problems by using a continuous model with an observer, so the behaviors can be tracked more efficiently, and details of dynamic behavior are not lost in quantization [9–13, 52, 70]. Quantitative methods also focus on eliminating the effects of noise, model uncertainty, and disturbances.

Diagnosis approaches for continuous systems deal with parametric faults, and may handle additive and/or multiplicative faults [12, 13, 52, 70]. Parametric faults may also be characterized as *abrupt*, i.e., sudden, discontinuous changes, or *incipient*, i.e., slow, smooth changes, and approaches deal with abrupt and/or incipient faults.

Diagnosis in continuous systems presents a number of challenges. Obtaining a precise model of the system is often very difficult. Therefore, model uncertainty will affect the performance of the diagnosis algorithm. Diagnosis must also be robust to sensor noise and disturbances.

Qualitative Diagnosis Methods

Qualitative methods use qualitative abstractions to model complex systems while still maintaining information that is important for diagnosis. The modeling language QSIM is developed to describe

continuous models qualitatively in [87]. Approaches that use QSIM to handle multiple faults in continuous devices are described in [73, 74]. The qualitative modeling framework quantizes the state space using landmark values and specifies qualitative relations between the quantized states, resulting in a set of qualitative differential equations. In a similar approach, the diagnostic system Livingstone models components as probabilistic transition systems, where component states are specified using qualitative constraints over small, finite domains [4]. Quantization, however, can result in a large number of states, which may be inefficient for large systems, much like the effects of quantization in timed DES.

Qualitative modeling techniques often yield relatively simple models that can describe behaviors that are key to effective diagnosis. However, techniques based on qualitative simulation may generate multiple possible behavior predictions because of the nondeterminism involved in the quantization of the state space. The approaches need to deal with these multiple predictions and eliminate those which are inconsistent or not physically possible, reducing their efficiency. Also, because the state space is quantized, some faulty behaviors may be hidden within what appears to be correct behaviors. That is, the changes in the model produced by faults may not change the qualitative values of the states, therefore, faults can be missed completely, or, fault detection can be delayed.

An approach that deals with the above problems is using qualitative deviation models, motivated by the fact that in many practical settings, the type of qualitative information that is available or useful for diagnosis is in reference to nominal values [88–90]. Information about failure modes is in the form of observations being too low or too high with respect to some nominal reference. Qualitative information is exploited to perform diagnosis of a complex system without continuous simulation in [89]. Qualitative deviation models with temporal information can also be utilized. A technique using process algebras is developed in [91]. However, the approach focuses more on characterization of diagnosis under the process algebra framework and much less on algorithms to solve the problem under the characterization, and process algebras can also be very cumbersome as a systems modeling language.

The qualitative deviation techniques are, in general, not very robust in most situations due to sensor noise. Also, for diagnosing dynamic systems, comparison to a fixed reference value is not enough. A proper comparison for fault detection must take into account the time-varying nature of the system measurements, which needs to be generated using a dynamic model of system behavior.

The Magellan-MT diagnostic engine is developed for the purpose of online diagnosis of dynamic systems in [71]. Knowledge about the system is encoded in the form of qualitative models. Both the monitoring and diagnosis stages use these qualitative models, which, because they employ a

quantization of the state space, only require new predictions to be made when a qualitative value of a measurement changes. Faults are described in qualitative terms and diagnosis begins when significant deviations are discovered. For the faults described in [71], information about the transients is not needed, so constraints which go across time are not needed either. Therefore, Magellan-MT simply performs static diagnosis at different time points in the system, based on new steady-state values.

Performing static diagnosis at different time points is not sufficient in most systems, so an extension of the QSIM language [87] is developed for model-based monitoring and diagnosis in [72]. The algorithm exploits hierarchy in the model to keep diagnosis efficient by using a constraint propagator and a dynamic model zooming strategy.

Effective diagnosis often requires knowledge of the fault transient, especially for faults related to energy storage elements, where diagnosis based simply on new qualitative steady-state values has little or no discriminatory power [20]. The TRANSCEND approach addresses this problem [20,21,45]. TRANSCEND is different from other qualitative approaches in that it monitors the system in the continuous domain, using an observer, and performs fault isolation in the qualitative domain based on deviations from nominal behavior. In contrast to traditional qualitative deviations models that use only the steady-state values to diagnose, diagnosis in TRANSCEND is based on the transients produced by faults. The TRANSCEND approach has also been extended to hybrid systems [14,36]. Since TRANSCEND forms the basis for the research of this dissertation, it will be described in greater detail in Chapter III.

Quantitative Diagnosis Methods

Quantitative diagnosis methods for continuous systems exploit the functional redundancy between the model and the sensor measurements [12, 13, 52, 70]. Relations over observable variables are computed based on the system equations and the available sensors, where each relation is designed to implicate a subset of possible faults. These relations produce *residuals* through residual generation methods. Quantitative methods often include some type of design procedure to determine the relations systematically.

Many diagnosis methods for continuous systems are based on *analytical redundancy relations* (ARRs). Essentially, an ARR is a constraint of the form $0 = r$, where r, the residual, is a function of only the observable variables of the system [92]. If an ARR is violated, it indicates that a fault is present. As an example, consider a resistive fault in a circuit. A possible ARR is $v_R(t) - i_R(t)R = 0$, where R is the resistance, $v_R(t)$ is the voltage across the resistor, and $i_R(t)$ is the current

Table 1: Example Structure Matrix for ARRs

Fault	r_1	r_2	r_3
f_1	0	1	1
f_2	1	0	1
f_3	1	1	1

through the resistor. Assuming there are no sensor faults, this ARR will only become significantly nonzero if the value of R is not the true value. ARRs are generated through a design procedure that eliminates the unobservable variables in the system equations. This design procedure can be complicated for complex systems, especially those with nonlinear behaviors and multiplicative faults (see, e.g., [9, 93, 94]).

Parity relations are a form of ARRs [9, 10]. With parity relations, a transformation of the input-output form of the system equations is derived to obtain the desired form of the residual structure matrix, which is designed to enhance fault isolation. ARRs are designed to only respond to a subset of the faults such that if a fault occurs, the set of nonzero residuals directly identifies that fault. Table 1 provides an example of a structure matrix. The residuals are given by r_i and the faults by f_i. Different combinations of nonzero residuals point to different faults, e.g., if f_1 occurs, r_2 and r_3 will be nonzero. In practical situations, a threshold is applied, and fault detection looks for the residuals crossing these threshold values before using them for isolation. With parity relations, the design process is algebraic for linear systems with additive faults, however, the approach becomes very complex for nonlinear systems and multiplicative faults, and only applies under certain restrictive assumptions.

Structural properties leading to residual structures that are robust to errors in fault detection are described in [10]. For dealing with multiple faults under this paradigm, a structure matrix must be designed with special properties. The most effective structure is a diagonal matrix, because it enables unique identification of all possible combinations of faults. Since each residual responds to exactly one fault in a diagonal structure, then every nonzero residual implicates a single fault, so a set of nonzero residuals implicates a set of faults. However, such a structure is often hard to achieve. Other structures are possible, but are limited in the maximum number of simultaneous faults [10].

Residual generators must be designed to be robust to noise and model uncertainty. Simply applying a threshold to the residuals is typically not enough. Observer-based fault detection and isolation is discussed in [11]. Observers estimate the system trajectory based on the system model and the sensor measurements. In deterministic settings, a Luenberger observer can be used, and in a stochastic setting, a Kalman filter can be used [11]. The residual is computed using estimated

outputs. With observer-based approaches, diagnosing only sensor faults is straightforward, because $\hat{y}_i(t) - y_i(t)$ would only be nonzero if sensor i was faulty. Conditions for robustness of observer-based residual generators are described in [11].

A crucial problem arises with ARRs for continuous dynamic systems with multiplicative faults. When a fault occurs, it is not likely that all residuals fire at the same time. If, for example, f_3 occurs in the example of Table 1, it could be that r_2 and r_3 fire quickly, but r_1 may not fire as quickly. Therefore, the system may isolate f_1 instead of f_3. This issue is addressed by the dynamic table of states (DTS) method [24]. A symptom detection time, associated with each residual, defines the maximum amount of time from the occurrence of any fault to the symptom appearance [24]. Minimum times are also considered in [23]. Using this information, the approach overcomes the pitfall in assuming near-instantaneous residual firings. Obtaining this kind of timing information, however, may be difficult for complex systems. And, unless the fault magnitudes are bounded, upper bounds on symptom appearance cannot be computed. Since these times have to correspond to maximum times for *any* fault, they are also overly conservative because different faults may have different maximum times [25].

Different methods for including timing information for fault isolation are considered in [25]. One method is to set a maximum waiting time for symptom appearance as in [23, 24]. Another method is to consider the order of symptom appearance in what is called a *dynamic fault signature matrix*. For some faults, symptoms appear in different temporal orders, therefore, they can be isolated by considering this information [25–28]. A fault diagnosis algorithm that uses temporal orders is described in [26]. Results illustrate that the diagnostic precision is improved over methods that do not use timing information, and diagnostic error is improved over methods like DTS. Since the approach is based on ARRs, it is difficult to apply the approach to nonlinear systems and multiplicative faults. Also, how to systematically generate the temporal orderings is not addressed. Similar techniques that do provide a way to compute the temporal orderings are developed in this dissertation and described in Chapters V, VI, and VII. These techniques have also been reported in [27, 28, 38, 39, 82, 86, 95, 96].

Discussion

Most qualitative methods use a qualitative model for system monitoring, thus do not achieve fast fault detection, and have to deal with extraneous qualitative behaviors that are generated but do

not match the system behavior. Reasoning based on qualitative deviations models is efficient, but a continuous observer is needed to track the system and achieve fast fault detection.

Quantitative, ARR-based approaches have been applied successfully in many areas, but lack some important features. First, the design procedure for the residual generators does not easily generalize to multiplicative faults and nonlinear systems. Newer approaches recognize the importance of delayed effects in the system [23–26], however the use of time bounds can introduce errors when faults outside the considered magnitude bounds are encountered.

ARR approaches also do not provide sufficient discriminatory power, because only a binary test is performed on the residual. In many practical cases, the sign of the residual can also help to discriminate faults, as well as information on the transient behavior produced by the fault [20].

Continuous systems approaches cannot be directly applied to hybrid systems, because they do not account for discrete mode behavior. Therefore, they cannot address the change in system dynamics due to a mode change, or faults associated directly with mode changes. ARRs can be extended to hybrid systems in a straightforward way by designing residual generators for each mode [35]. However, such an approach does not scale to systems with a large number of modes.

The diagnosis approach described in this dissertation uses observer-based residual generation, where there is a one-to-one mapping between residuals and measurements. System tracking, therefore, takes place in the continuous domain, but diagnostic reasoning is performed in the qualitative domain using deviations from nominal behavior. The sign and the dynamics of the deviation are used, as well as the temporal order in which deviations are observed, which increases diagnostic efficiency and precision.

Hybrid Systems Diagnosis

While logic-based, DES, and continuous systems diagnosis methods have been successful, the methods do not extend to complex systems that exhibit hybrid behaviors [16, 97]. Since hybrid systems mix continuous and discrete behavior, they need to combine methods from continuous and discrete-event systems diagnosis. That is, they must reason about differences in observer-estimated continuous behavior in a mode of operation, and system events such as mode changes and symptom appearance that may be attributed to faulty modes.

Hybrid automata are a general scheme for hybrid system modeling [98]. In this modeling paradigm, hybrid system behavior is defined by the continuous system dynamics and discrete mode changes that alter the continuous dynamics and may reset state variables. Mode transitions are

controlled or autonomous. Controlled transitions depend only on external events, and autonomous transitions depend on internal system variables. Transition guards define when the mode transitions are enabled. In hybrid systems, faults can affect the continuous behavior within a mode or the discrete mode behavior, i.e., they may be parametric or discrete. Faults may be modeled in hybrid systems by introducing parameters into the system model or by explicit fault events and fault modes [19, 37, 99].

Example. Consider the switched circuit given in Fig. 5. Its hybrid automaton model is given by Fig. 6 (note the similarity to Fig. 3). We assume that the switch Sw_1 responds to external control events *Open* and *Close*. The special failure events *Fail Open* and *Fail Closed* and their corresponding failure modes model a switching fault, which changes the configuration of the system. There are four modes modeling open, closed, stuck-open, and stuck-closed states of the switch. Selecting the state variables as $i_1(t)$, the current through L_1, and $v_2(t)$, the voltage through C_1, and the outputs as $i_1(t)$, $v_2(t)$, and $i_3(t)$, the current through R_2, the system equations for the open modes are given by f_1 and g_1:

$$\begin{aligned} f_1(t, \mathbf{x}(t), \mathbf{u}(t)) &= \begin{bmatrix} -\frac{R_1}{L_1} & -\frac{1}{L_1} \\ \frac{1}{C_1} & 0 \end{bmatrix} \begin{bmatrix} i_1(t) \\ v_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_1} \\ 0 \end{bmatrix} v(t) \\ g_1(t, \mathbf{x}(t), \mathbf{u}(t)) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & \frac{1}{R_2} \end{bmatrix} \begin{bmatrix} i_1(t) \\ v_2(t) \end{bmatrix}. \end{aligned}$$

The system equations for the closed modes are given by f_2 and g_2:

$$\begin{aligned} f_2(t, \mathbf{x}(t), \mathbf{u}(t)) &= \begin{bmatrix} -\frac{R_1}{L_1} & -\frac{1}{L_1} \\ \frac{1}{C_1} & -\frac{1}{C_1 R_2} \end{bmatrix} \begin{bmatrix} i_1(t) \\ v_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_1} \\ 0 \end{bmatrix} v(t) \\ g_2(t, \mathbf{x}(t), \mathbf{u}(t)) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} i_1(t) \\ v_2(t) \end{bmatrix} \end{aligned}$$

A discrete fault such as a switching failure in this system can be diagnosed using DES methods as described previously. However, to properly handle parametric faults, we must also bring in techniques from continuous systems diagnosis. In hybrid systems diagnosis, the transition from the open mode to the stuck-closed mode can be detected immediately, because the sensor measurements

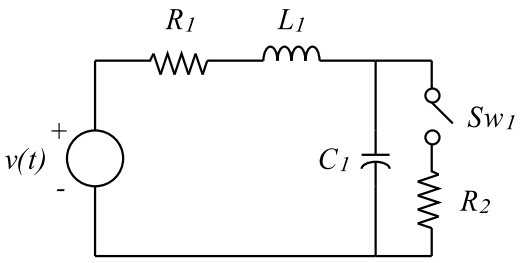


Figure 5: Example switched circuit.

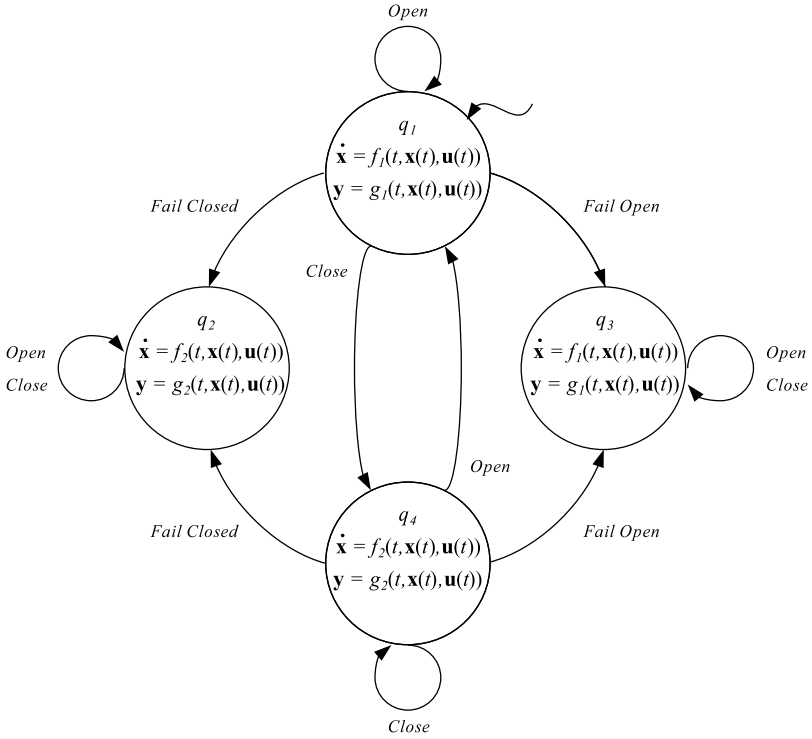


Figure 6: Hybrid automaton for the switched circuit.

are monitored frequently as in continuous systems diagnosis. In a pure DES approach, we would have to wait for the *Close* command to be issued before the fault could be detected. Additionally, autonomous transitions would be unobservable in the DES framework, but the possible autonomous transitions can be limited by correctly observing the continuous behavior of the system.

Due to the complexity of the behaviors modeled by hybrid systems, a number of challenges arise in diagnosis. In order to detect faulty behavior, the system state must be tracked. As in continuous systems diagnosis, the best way to achieve this is with an observer. Building hybrid observers is a challenge in itself. Multiple modes may have to be tracked simultaneously if the system model is nondeterministic, or if there is uncertainty about whether autonomous transitions have been taken

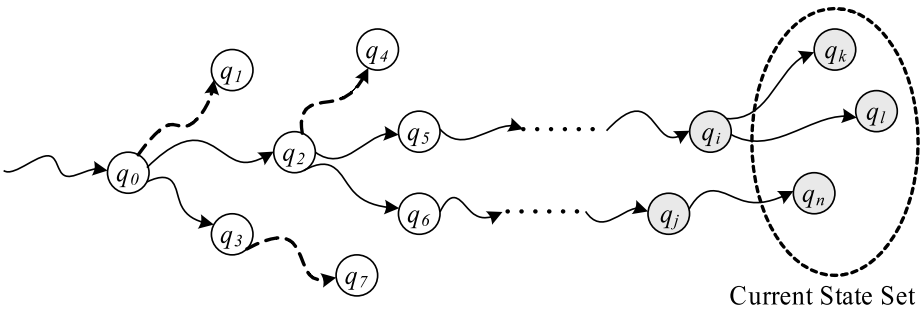


Figure 7: Mode estimation.

or not. Further, different types of faults can manifest in hybrid systems, therefore, a robust hybrid diagnosis scheme should be able to handle all fault types.

Discrete Fault Approaches

Since many diagnosis techniques rely on estimating the system behavior, estimation in hybrid systems is a fundamental issue to hybrid system diagnosis. Some hybrid diagnosis techniques, in fact, are based purely on mode estimation. In these approaches, only discrete faults, modeled as fault events or explicit fault modes, are handled. Therefore, there must exist a mode in the system model for each fault considered. If the current mode estimate is a fault mode, then the fault is detected and isolated. This is the general idea behind approaches based in logic [4, 73, 74] and DES [5–8, 64, 68]. In these approaches, estimating the system state solves the diagnosis problem. The difference in the hybrid case is the use of hybrid models. Hybrid system diagnosability is investigated in this framework in [76, 77].

The estimation approach is illustrated in Fig. 7. The system starts in some known initial mode q_0 (there may be an initial state set as well). As the system is tracked, new modes are generated (e.g., $q_1, q_2, q_3, \ldots$). Because the set of possible states explodes exponentially, only the most probable trajectories are tracked. For example, q_1, q_4, and q_7 are dropped from consideration. At some current set of states, e.g. q_i and q_j, new modes are hypothesized to form the new current state set. The states will then be probabilistically evaluated according to their consistency with the observations.

Robust observers with bounded estimation error for uncertain nonlinear systems are constructed in [34]. The hybrid observer consists of both a mode observer and a continuous observer, with a Kalman filter for each mode. Faults are modeled as discrete modes, so identifying the correct mode isolates the fault. Using multiple Kalman filters, however, results in high space requirements for

large hybrid systems. Further, the results presented in [34] show slow mode tracking, therefore leading to delayed fault detection and isolation.

The problem of mode estimation is tackled in [30]. The system is modeled as concurrent probabilistic hybrid automata that combine hidden Markov models with continuous dynamics, where the mode transitions are probabilistic. The continuous dynamics are tracked using a bank of extended Kalman filters. Each filter tracks a possible mode of the system as determined by a hidden Markov observer. The hybrid observer decides which modes are most likely, and only a set of the most likely modes are tracked using the Kalman filters. Since most of the probability space can be covered by just a few modes, this is more efficient than using a filter for every mode, and achieves a close enough approximation.

The hybrid estimation scheme developed in [30] is extended in [15] to handle unknown behavioral modes that are common in discrete logic-based approaches like GDE [3]. In unknown modes, there are no continuous dynamics associated with the mode so there are no constraints on the continuous evolution. When an unknown mode is entered, it is tracked in a degraded fashion by utilizing knowledge of the system structure. The motivation behind tracking unknown behavioral modes is, for example, subtle component degradations which can only be modeled by introducing a large number of modes to account for the possible rates of degradation [15]. However, adding the unknown modes does not help in isolating these types of faults, because it only results in the diagnoser discovering that something unknown has occurred. Further, the tracking in those modes is severely degraded.

Using multiple Kalman filters can be computationally expensive, so an alternative approach is to use particle filtering methods. With particle filters, the computational complexity is a direct function of how many particles are used in the estimation. In addition, particle filters, unlike Kalman filters, are not restricted to making the Gaussian noise assumption. Particle filters are used to estimate the discrete mode of the system and its continuous outputs in [16, 33]. The efficiency of the hybrid estimation scheme is improved by modifying transition guards. A small constant is added to the guard boundaries to avoid chattering between mode estimates. This improves the performance of the estimation at points close to autonomous transitions. Performance of the estimation algorithm is improved by increasing the number of particles.

Particle filters for hybrid system diagnosis are developed and applied to planetary rovers in [31]. The authors point out, however, that particle filtering cannot be directly applied for diagnosis, because fault states typically have very low transition probabilities, therefore, there may not be enough particles in the true fault mode to produce the mode as a likely current mode. This can be

solved by using more particles as in [33], but this increases the computation time and, hence, is not practical for systems with limited resources. Using importance sampling, the approach ensures that there are always enough particles in the possible fault modes without biasing the results [31].

The main disadvantages of approaches that are based solely on mode estimation are (*i*) they handle only discrete faults, and (*ii*) fault detection may lag or faults may be missed because fault modes have very low probabilities. Importance sampling is one method to overcome the second issue, but it does not provide a basis for selecting which modes are important. Selecting all possible fault modes increases the computation requirements. Many newer techniques use a reasoner to decide which fault modes are consistent with observed behavior, and these form the set of considered modes [32, 75]. A key advantage of using a reasoner is that it can provide fault hypotheses very quickly without performing any estimation, because the reasoning is done in some abstract domain.

Estimation of uncertain hybrid concurrent systems is presented in [75]. Diagnosis of faults that change the system mode is performed using a consistency-based approach. Particle filters for mode estimation and the reasoner Livingstone 3 (L3) are combined in [32]. Faults are modeled as unobservable transitions to faulty modes of the system. The basic diagnosis approach is illustrated in Fig. 8. A particle filter is used to track the system to produce a nominal mode estimate and detect faults. Look-ahead is used to properly account for the low probability states. When a fault is detected, the true system mode is unknown (denoted by the $q_?$ mode). The L3 diagnosis engine produces possible consistent candidates (e.g., q_{f1}, q_{f2}, and q_{f3}) using the observations, and these possible faults are tracked to determine which has occurred in the system, so only discrete faults are handled. Limiting assumptions about the occurrence of multiple faults are also made. Further, the approach assumes that faults can be detected within some known delay after they occur. For faults represented as modes, this may be reasonable, but for many faults the detection delay will depend on fault magnitude, as well as sensor noise, model uncertainty, and other factors. The Hybrid Diagnosis Engine extends this work to a general modeling and diagnosis framework for hybrid systems [18]. Models can be specified at different levels of abstraction, supporting different types of diagnosis algorithms.

Parametric Fault Approaches

Other approaches focus on parametric faults, which directly affect the continuous dynamics within a mode. Hybrid systems diagnosis using TRANSCEND [20] as a reasoner is described in [29]. In contrast to the previous hybrid diagnosis approaches, this work considers only single, abrupt, para-

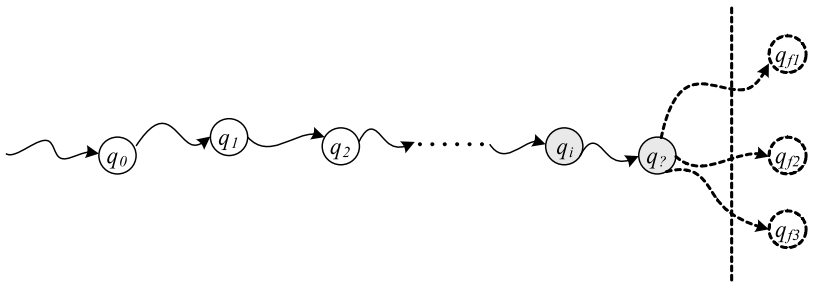


Figure 8: Hybrid diagnosis based on mode estimation with a reasoner.

metric faults. The system is tracked using a hybrid observer, and it is assumed that there are no autonomous mode changes. When the system behavior deviates outside of what is predicted by the model, fault candidates are hypothesized using TRANSCEND. A model is tracked for each of these possible candidates using parameter estimation and model fitting. The approach is extended by formulating the problem as Bayesian model selection in [29]. The possible candidates determined by the qualitative diagnosis are used to bias the temporal prior, so the posterior for the fault models will be weighted more heavily. To handle non-Gaussian noise and nonlinear dynamics, a particle filter is utilized, which also has the benefit of being able to simultaneously track the different models. Importance sampling could also be used to do this, as in [31], but using the results from the qualitative diagnosis helps to determine which fault models are important to track, rather than relying solely on a priori fault probabilities. This is also similar to the use of L3 in [32].

TRANSCEND is extended to hybrid systems in [14, 22, 49]. Assuming single, abrupt, parametric faults, estimation is performed using a hybrid observer implemented as a switched extended Kalman filter, and fault isolation is based on analysis of fault transients manifesting as deviations from nominal behavior. Both controlled and autonomous mode changes are considered, and extended diagnosis algorithms for hybrid systems are presented. Hybrid TRANSCEND, since it forms a basis for the results of this dissertation, will be described in more detail in the next chapter.

Combined Approaches

Very little work has addressed a combined approach to diagnosis of both parametric and discrete faults in hybrid systems. An approach that models the system as a composition of hybrid automata is described in [19]. The system model is constructed to satisfy certain constraints, allowing it to be abstracted to a timed Petri net model that is used to monitor the system and detect faults. When faults are detected, a decision tree diagnoser is invoked. The diagnoser is constructed offline based on pre-generation of fault symptoms using the detailed hybrid system model in simulation.

Pre-generating fault symptoms via simulation may be difficult for large systems with many modes because different system modes may produce different qualitative symptoms [14]. The diagnoser uses a mode estimation algorithm to request sensor tests or uses information provided from the Petri net, assuming that sensor faults do not occur.

A combined approach based on parity relations is described in [35]. Parity relations are generated for identifying both parametric faults and discrete modes, i.e., a residual r_{ij} is zero if the mode is i and fault j is not present. The set of nonzero residuals thus identifies both the correct mode of the system and whether some fault has occurred. As with other parity relations approaches, this approach is difficult to apply to nonlinear systems with multiplicative faults. Since a set of parity relations is computed for each mode, the approach may not be feasible when the number of modes is very large.

Discussion

Many hybrid systems approaches handle only parametric or only discrete faults, and the approaches that handle both do not scale well to large systems. Further, approaches make the single fault assumption, or make strict assumptions about the occurrence of multiple faults. In purely estimation-based approaches for discrete faults, the number of possible multiple-fault modes becomes exponential in the number of considered faults. Although the probability of multiple faults occurring is low, diagnosers that make the single fault assumption may generate incorrect diagnoses when multiple faults occur. Because of the disadvantages presented by the combined approaches to hybrid systems diagnosis, an efficient scheme in a combined framework is needed. In this dissertation, we develop an approach that addresses both parametric and discrete faults in an integrated framework, based on qualitative reasoning over deviations from nominal behavior.

Summary

As continuous systems diagnosis methods have shown, the best way to track a system is using a continuous- or discrete-time representation. Logic-based and DES diagnosis approaches which rely on abstracted models that quantize the state space lag in fault detection and may miss crucial faults that are hidden within the quantized state trajectory. They are also limited in the classes of faults they can easily address. Continuous systems approaches, unlike DES approaches, do not handle event-based information, like time of symptom appearance, in a robust way. Additionally, ARR approaches are difficult to apply to nonlinear systems and multiplicative faults, and the design

procedure for residual generation is complex. Multiple faults are also ignored or difficult to design for in both ARR approaches and many hybrid diagnosis approaches. Further, many hybrid systems diagnosis approaches restrict their faults to exclusively parametric or discrete faults.

In the following chapters, we describe methods for multiple fault diagnosis in hybrid systems based on TRANSCEND and Hybrid TRANSCEND. Like many continuous systems approaches, an observer is used to track system behavior and provide a nominal reference for diagnosis. Fault isolation is performed at a qualitative abstraction, and we use event-based techniques from the DES area to improve fault isolation based on observed sequences of measurement deviations from nominal behavior. Unlike many hybrid systems approaches, we diagnose both parametric and discrete faults. Multiple faults are also handled by using concepts from the traditional logic-based approaches.

CHAPTER III

HYBRID SYSTEMS DIAGNOSIS: PROBLEM FORMULATION AND ARCHITECTURE

Most real world systems involve the interaction of physical and computational processes, therefore, they combine continuous-time and discrete mode behavior, and are modeled as *hybrid systems*. A hybrid system evolves in time in a continuous manner, but the continuous dynamics may change when the system mode changes. Because hybrid systems combine continuous and discrete behaviors, the diagnosis task becomes very challenging. Our approach extends the TRANSCEND [20, 45] and the Hybrid TRANSCEND [14, 22] methodologies, which diagnoses faults based on the transients they cause in the system behavior immediately after they occur.

Hybrid TRANSCEND addresses fault isolation in hybrid systems for single, abrupt, parametric faults [14]. Other than mode changes, it does not use any event-based information in fault isolation, and does not address discrete fault diagnosis or multiple fault diagnosis. This dissertation extends Hybrid TRANSCEND to address the above challenges. We develop an event-based framework that (*i*) utilizes temporal orders of measurement deviations to improve the discriminatory ability of the diagnosis algorithms, (*ii*) addresses both parametric and discrete faults, and (*iii*) isolates multiple-fault candidates.

In order to develop the new framework, we first present a general formulation of the diagnosis problem for hybrid systems. We then describe the challenges this problem poses and specify the assumptions we make. We then present the TRANSCEND-based diagnosis architecture for fault diagnosis of hybrid systems.

Problem Formulation

We consider the problems of single and multiple fault diagnosis in hybrid systems. Formally, *hybrid systems* can be defined as follows [19, 98, 99].

Definition 1. A *hybrid system* is defined as $H = (Q, X, \Sigma, Init, E, f, G)$, where Q is the set of discrete states, X is the continuous state space, Σ is the set of transition events, $Init \in Q \times X$ is the initial state, $E \subseteq Q \times \Sigma \times Q$ is the set of discrete transitions, $f : \mathbb{R} \times Q \times X \rightarrow X$ is the flow condition, and $G : E \rightarrow 2^X$ is the transition guard function.

Each mode of the system defines a new set of continuous dynamics. Transitions between modes can occur due to external and known controller actions such as controlled relay switching, or autonomous behaviors that depend on internal system variables such as circuit breakers tripping or discrete faults.

The most general problem we consider is multiple fault diagnosis in hybrid systems, so we formulate the problem for this case, which reduces to simpler cases (i.e., single fault diagnosis and continuous systems diagnosis) when additional assumptions are made about the system dynamics or cardinality of the faults. Given a hybrid system, we first define a set of faults, $F = \{f_1, f_2, \ldots, f_n\}$, which may be either parametric or discrete. Second, we define a set of measurements $M = \{m_1, m_2, \ldots, m_p\}$, which are time-varying signals obtained from the available sensors. Last, we denote the set of modes as $Q = \{q_1, q_2, \ldots, q_r\}$. We refer to a set using uppercase symbols and an element of the set using lowercase symbols.

Diagnosis occurs over a time interval. We denote the time of fault occurrence as t_f and the time of fault detection as $t_d \geq t_f$. Diagnosis begins at t_d. Mode changes may occur after t_f, and faults may affect the measurements differently in different modes of operation, so possible mode changes need to be accounted for [14]. We therefore adopt the following definition of a *candidate*.

Definition 2 (Candidate)**.** A *candidate* c is defined as $c = (F_i, q_i)$, where $F_i \subseteq F$ is a set of faults, and $q_i \in Q$ is a mode. The set of all candidates is denoted as C.

Example. The candidate $(\{f_1, f_2\}, q_0)$ implies that both f_1 and f_2 have occurred and the system is currently in mode q_0. In shorthand, we represent this candidate as $(f_1 f_2, q_0)$.

We obtain observations from the sensors, and find candidates *consistent* with these observations. A candidate is consistent with the observations if, when introduced into the system model, the model-generated behaviors match the observations. Since consistency depends on the model used for diagnosis, its definition can vary among the different approaches. A *diagnosis* is a collection of candidates that are consistent with the observations provided to the diagnoser.

Definition 3 (Diagnosis)**.** At time $t \geq t_f$, a *diagnosis* $d \subseteq C$ is a set of candidates consistent with the observations made on the system during the interval $[t_f, t]$.

Example. The diagnosis $\{(f_1 f_2, q_0), (f_3, q_1)\}$ means that two candidates are consistent with the observations made up to the current point in time. Either both f_1 and f_2 have occurred and the system is in q_0, or f_3 has occurred and the system is in mode q_1. The diagnosis $\{(f_1, q_0), (f_1 f_2, q_0), (f_2 f_3, q_0)\}$ means that the predicted system mode is q_0, and either f_1 has occurred by itself, f_1 and f_2 both occurred, or f_2 and f_3 both occurred.

We wish to find *all* possible consistent candidates, i.e., the *maximal diagnosis*.

Definition 4 (Maximal Diagnosis)**.** At current time $t \geq t_f$, the *maximal diagnosis* is the set of all candidates consistent with the observations made on the system during the interval $[t_f, t]$.

We desire the maximal diagnosis because it gives us every possible explanation for the faulty behavior. This leads us to the following problem definition.

Problem (Multiple Fault Diagnosis in Hybrid Systems)**.** Given a hybrid system $\mathcal{S}$ with a set of faults, F, a set of measurements, M, a set of modes Q, and a candidate cardinality limit l, the *multiple fault diagnosis problem* for hybrid systems is to find the maximal diagnosis d of candidates with cardinality $\leq l$ for a given sequence of observations.

The problem definition is the most general and applies to single fault diagnosis or continuous systems diagnosis. For example, if we take Q to be a singleton, and $l = 1$, then we are performing single fault diagnosis in continuous systems. Candidates become single faults, and diagnoses become sets of faults.

Challenges in Hybrid Systems Diagnosis

In addition to the challenges in diagnosis of dynamic systems presented in Chapter II, additional issues arise in trying to solve the diagnosis problem for hybrid systems. Next, we identify these challenges and describe the assumptions that make the problem more manageable.

Finding the Maximal Diagnosis

Finding the maximal diagnosis is a hard problem. The space of possible candidates is exponential in the number of faults, therefore, space and time complexity of diagnosis algorithms becomes exponential in the general case. Therefore, the maximal diagnosis is not useful in practice. We can simplify this problem by adopting the idea of *minimality* [3, 57] and extending it to hybrid systems.

Definition 5 (Minimal Diagnosis)**.** A diagnosis d is *minimal* if for all $(F_i, q) \in d$, there does not exist some $(F_j, q) \in d$, such that $F_j \subset F_i$.

This definition of minimality means that for a given mode, we cannot have two candidates for that mode in the diagnosis, such that the fault set of one is a superset of the other. The one which is a subset is the simpler, more probable explanation and *covers* the superset. For example, the diagnosis $\{(f_1, q_0), (f_1 f_2, q_0)\}$ is not minimal because $\{f_1\} \subset \{f_1, f_2\}$ and both candidates predict

the system is in mode q_0. The diagnosis $\{(f_1, q_0), (f_1 f_2, q_1)\}$ is, however, minimal, because although $\{f_1\} \subset \{f_1, f_2\}$, the candidates are for different modes.

In dynamic systems, it is necessary to abstract the continuous stream of measurements to discrete observations, since it is difficult to reason directly with the complete data stream. Because information is lost, fault masking may occur, and therefore, we can only be certain about minimal diagnoses. For example, consider a system with two possible faults, f_1 and f_2, that can occur. Assume that fault f_1 occurs. It will affect the system and change the outputs. Assume that a second fault, f_2 occurs. It will also change the system outputs. If f_1 has a much larger magnitude than f_2, the effects of f_2 may never be observed given the set of measurements and the chosen level of abstraction, i.e., f_1 completely masks f_2. In this case, we would obtain a diagnosis of only f_1. But, the candidate $\{f_1, f_2\}$ is also consistent, it is just that the behaviors due to f_2 were never directly observed. Therefore, a maximal diagnosis would contain both $\{f_1\}$ and $\{f_1, f_2\}$ as candidates. It would not contain $\{f_2\}$, because f_2 occurring by itself is ruled out. Representing just the minimal diagnosis, that containing only f_1, is sufficient to represent the complete set of consistent candidates. Therefore, for some consistent fault set F_i, any $F_j \supset F_i$ is also consistent. This also applies to static systems [2, 3, 57]. The key advantage of a minimal diagnosis, then, is that it is a concise representation of the maximal diagnosis, so finding the minimal diagnosis, which is much more efficient and useful in practice, gives us the maximal diagnosis. We do not have to explicitly compute all possible supersets because they are implied by the minimal diagnosis.

Due to fault masking, we can only adopt a *best effort* paradigm to diagnosis. That is, if multiple faults occur, then, using a discrete abstraction of the continuous measurements, we cannot always determine with certainty all the faults that did occur. What the minimal diagnosis represents is the possibilities that have been determined with certainty. So, a diagnosis of $\{f_1, f_2 f_3\}$ would mean that we know with certainty that either f_1 must have occurred, or f_2 together with f_3 must have occurred, and also that f_2 or f_3 could not have occurred by themselves. If, for example, we see effects that are consistent with f_1 occurring by itself, we cannot rule out the possibility that f_1 occurred with some other fault that it masked.

Like logic-based approaches to model-based diagnosis, we exploit the property of minimality in our diagnosis algorithms. Candidates with more faults are only explored as possible candidates when candidates with fewer faults can no longer explain the observations. Also, we prune the diagnosis to make it minimal. For example, if the diagnosis is $\{(f_1, q_0), (f_2, q_0)\}$, and a new observation makes (f_1, q_0) inconsistent, we then consider larger candidates containing f_1. This may result, for example, in a new diagnosis of $\{(f_2, q_0), (f_1 f_2, q_0), (f_1 f_3, q_0)\}$, which would then be pruned to

$\{(f_2, q_0), (f_1 f_3, q_0)\}$. Pruning $(f_1 f_2, q_0)$ will not affect future minimal diagnoses, because if we obtain another observation directly implicating f_1, we will regenerate the candidate.

While the minimal diagnosis does provide a concise representation of the maximal diagnosis, it still grows exponentially with the number of discrete observations [3]. Therefore, we make the following assumption.

Assumption 1 (Candidate Cardinality)**.** At most l faults occur together in the system.

Under this assumption, each candidate contains at most l faults, so the candidate space is polynomial in l. Note that by faults occurring together, we mean that when the first fault occurs, not all symptoms are detected before the next fault occurs. Otherwise, the problem reduces to single fault diagnosis.

Handling Mode Changes

After a fault occurs at time t_f, it is detected at some later time $t_d \geq t_f$. Therefore, we do not know the true time of fault occurrence, and, hence, do not know the system mode at that time, $q(t_f^-)$. Thus, we must hypothesize, based on the system model, which autonomous mode changes may have occurred during the time interval $[t_f^-, t_d]$, and which mode changes may occur after t_d [14, 22].

Mode changes can be hypothesized either based on controlled or autonomous switching. Controlled mode changes are known, but autonomous mode changes must be hypothesized. For example, a controlled mode change could be a commanded on or off action of a relay in an electrical system. Circuit breakers turn off based on an endogenous variable, i.e., the current passing through them, therefore, the mode change when a circuit breaker trips is autonomous.

Like [14, 22], we make an assumption on the number of mode changes that occur between fault occurrence and detection.

Assumption 2 (Number of Mode Changes Before Fault Detection)**.** The actual system mode at t_f^- is within n_{back} mode changes from the estimated mode at t_d.

Under this assumption, we limit the number of possible modes of fault occurrence. Therefore, diagnosis is more efficient because there are a limited number of mode trajectories after fault occurrence to consider.

We also assume that mode changes do not cause resets in state variables.

Assumption 3 (No State Resets)**.** Mode changes do not cause state variable resets.

State resets can be handled within our framework, as long as the effects can be qualitatively propagated to determine the measurement deviations that they will cause. This has been described in [14, 22], therefore we do not present the details and refer the reader to those references.

Handling Different Fault Types

Faults may differ in whether they affect the continuous behavior or the discrete behavior of the system [19, 35–37, 99]. Faults may be parametric, directly affecting the continuous behavior within a mode, or may be discrete, changing the system mode. We allow both parametric and discrete faults.

Faults may also have different time-varying profiles. They may be persistent, i.e., always faulty after t_f, or intermittent, i.e., alternately nominal and faulty after t_f. Intermittent faults are difficult to characterize and diagnose, and have mostly been dealt with at very high levels of abstraction [100–104]. We restrict the faults to be persistent.

Assumption 4 (Persistent Faults)**.** All faults are persistent.

For a parametric fault, the change in the parameter value over time may be arbitrary. Two commonly accepted profiles are *abrupt*, i.e., a sudden change in value, and *incipient*, i.e., a slow change in value. We restrict parametric faults to be abrupt. Incipient fault profiles are more difficult to deal with qualitatively and require a tighter integration with quantitative methods [78, 79, 105–107].

Assumption 5 (Abrupt Faults)**.** All parametric faults are abrupt.

Although abrupt faults may be restrictive, they are a good approximation for many practical faults in sensors, actuators, and the physical process. Even if diagnosis methods for incipient or intermittent faults are available, they should be complemented by diagnosis of abrupt parametric and discrete faults that may have catastrophic consequences if not detected and isolated quickly. Fault modeling will be described in greater detail in Chapter IV.

Diagnosis Architecture

With these assumptions, we adopt an architecture for fault diagnosis in hybrid systems based on the TRANSCEND [20, 45] and the Hybrid TRANSCEND [14, 22] methodologies. In TRANSCEND, diagnosis is based on analysis of measurement deviations from nominal behavior. Like qualitative deviations models [88], we reason about faults based on observed differences with respect to nominal behavior, only the nominal reference is changing with time, as shown in Fig. 9. This can be viewed as a

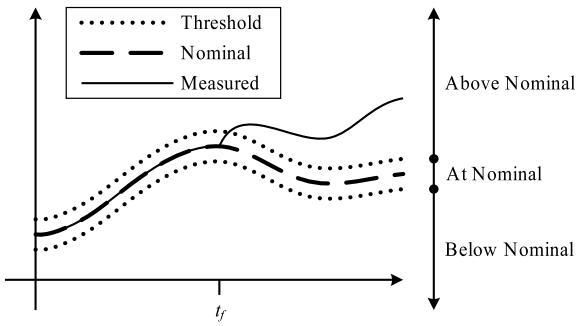


Figure 9: Fault isolation based on a qualitative abstraction.

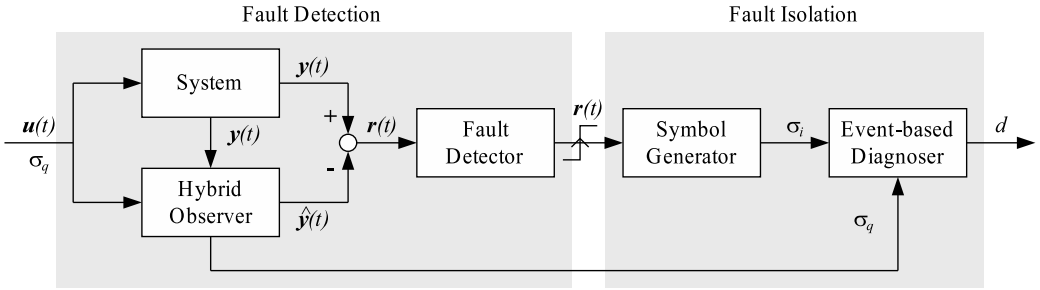


Figure 10: Event-based hybrid diagnosis architecture.

qualitative abstraction, as opposed to *quantitative* abstractions of the state space used by QSIM-based approaches [73, 74] and timed DES methods [68, 69]. The use of qualitative reasoning also makes fault isolation more computationally efficient than hybrid diagnosis approaches based on mode estimation, because they must simulate multiple possible trajectories.

The computational architecture is shown in Fig. 10. Since the system is dynamic, we continually track the system in discrete time using a *hybrid observer* to produce the nominal reference, based on known inputs and controlled mode changes. A *fault detector* determines if the measured values are above or below thresholded nominal values. Significant deviations are abstracted using a *symbol generator* as qualitative increasing/decreasing values. The symbol generator produces events representing the measurement deviations. The *event-based diagnoser* isolates faults based on measurement deviation events and controlled mode change events. Note that the diagnoser may be followed by a fault identification module, such as those described in [14, 22, 45].

The main contributions of this dissertation focus on the design and implementation of the event-based diagnoser for continuous and hybrid systems, which will be described in detail in Chapters V, VI, and VII. The other pieces of the architecture are briefly described in the rest of the section, and use the techniques developed in previous TRANSCEND work [14, 20–22, 28, 44–47, 49, 50]. However, they are extended to support both parametric and discrete faults. In particular, we extend the

modeling language to incorporate discrete fault events, and we extend the symbol generator to produce an additional symbol that is useful for discriminating between discrete and parametric faults.

System Model

We model physical systems using hybrid bond graphs (HBGs) [48]. HBGs support first principles, energy-based, physical systems modeling, and represent the hybrid dynamic behavior of a system. They are useful for diagnosis because they retain the system topology, enable the automatic derivation of system equations, and allow for the derivation of other representations that capture the causal and temporal relations between the system variables. We extend the HBG modeling language to incorporate discrete faults by introducing unobservable fault events into the local automata that control the switching between modes. HBGs and their extensions will be described in more detail in Chapter IV.

Hybrid Observer

Fault detection operates on the residuals, which are defined as the difference between the estimated and measured values of the output variables. For an ideal system with noiseless measurements and a perfect model, a nonzero residual indicates a fault occurrence. Noise and model imperfections make the residual generation and fault detection tasks more complex. We address this issue by using a hybrid observer to track the system trajectory, and defining the fault detection task as a statistical test of significance. The hybrid observer helps to compensate for sensor noise and modeling errors in producing the estimated behavior.

The hybrid observer can be implemented in many ways. Using the HBG modeling framework, we can automatically derive the system equations for any mode [48,108,109]. Using these equations, we generate an extended Kalman filter for the current mode of operation, accompanied by an automata scheme that implements the switching for the HBG [14,50]. The observer takes in the system inputs, $\mathbf{u}(t)$, controlled mode changes σ_q, and the sensor outputs, $\mathbf{y}(t)$, to produce the estimated outputs, $\hat{\mathbf{y}}(t)$, where t is discrete. In both the Kalman filter and the fault detection test, all noise is assumed to be zero-mean Gaussian. Kalman filters are very efficient and optimal for linear systems with Gaussian noise, which is a practical assumption for many systems. Alternatively, particle filters can be used, which can deal with non-Gaussian noise, but are less efficient [31–33].

Fault Detection

The difference between the observed outputs, $\mathbf{y}(t)$, and the estimated outputs, $\hat{\mathbf{y}}(t)$, define the residual vector $\mathbf{r}(t)$. A fault detector is associated with each available sensor. The fault detection scheme is based on a Z-test that uses the estimated variance of the residuals and a pre-specified confidence level to establish the significance of observed nonzero residuals. To cope with noise, we couple this with a sliding window technique to compute the mean and variance at different time points [44].

The Z-test is a statistical inference test that determines if a sample belongs to a population with a known distribution [110]. It requires the mean and standard deviation of the population, and the mean and size of the sample. These values are estimated using sliding windows over the residual for a measurement. A small sliding window (e.g., 5 samples), W_1, is used to estimate the current mean $\mu_r(t)$ of a residual $r(t) = y(t) - \hat{y}(t)$:

$$\mu_r(t) = \frac{1}{W_1} \sum_{i=t-W_1+1}^{t} r(i). \tag{1}$$

We take the mean of the population to be zero, since the residual should be zero when the system is free of faults. We compute the variance from data history of the nominal residual signal over a window W_2 preceding W_1, where $W_2 \gg W_1$ (e.g., 100 samples)[1], as an estimate of the true variance:

$$\mu'_r(t) = \frac{1}{W_2} \sum_{i=t-W_1-W_2+1}^{t-W_1} r(i) \tag{2}$$

$$\sigma_r^2(t) = \frac{1}{W_2} \sum_{i=t-W_1-W_2+1}^{t-W_1} (r(i) - \mu'_r(t))^2. \tag{3}$$

We perform a statistical hypothesis test for the residual, using the Z-test. After setting a confidence level (e.g., 95%), the Z-test provides bounds z^- and z^+. The thresholds $\varepsilon_r^-(t)$ and $\varepsilon_r^+(t)$ are computed based on these bounds and the z-score:

$$\varepsilon_r^-(t) = z^- \frac{\sigma_r(t)}{\sqrt{W_1}} + E \tag{4}$$

$$\varepsilon_r^+(t) = z^+ \frac{\sigma_r(t)}{\sqrt{W_1}} - E, \tag{5}$$

[1]To ensure that W_2 does not have any samples from after fault occurrence, the window can be shifted earlier by some fixed number of samples.

where E is a modeling error term. A mean value $\mu_r(t)$ that lies outside of the thresholds at time t implies a fault. Unlike techniques that use fixed thresholds, this strategy can be viewed as a dynamic thresholding technique, because the threshold is recomputed at each time step based on the estimated signal variance.

Still, the fault detectors must be tuned to optimize performance. Tuning a fault detector refers to setting its sensitivity. Optimally, each fault detector will be tuned to minimize missed detections (false negatives), and false alarms (false positives). Since we assume faults are persistent, missed detections are unlikely, unless the fault magnitude is very small. In that case, however, the system is still operating within normal ranges. A more sensitive fault detector will detect faults faster and have fewer false negatives, but may generate more false positives. Similarly, a less sensitive fault detector will take more time to detect faults and may produce more false negatives but fewer false positives. In practice, the fault detectors are empirically tuned to the highest possible sensitivity to avoid false alarms for the expected or observed levels of noise for the system under nominal conditions. The detectors can be tuned by adjusting window sizes, the confidence level, and the modeling error term [44]. The fault detection module is interchangeable and can be replaced. Other fault detection strategies can be found in [111, 112] and the references therein.

Symbol Generation

Once a fault has been detected, fault isolation is invoked. Detection of a fault indicates that some measurement has deviated from its nominal value. Symbol generation abstracts this deviation symbolically into qualitative increasing/decreasing values, which produces a measurement deviation event, σ_0, for the diagnoser. As further measurements deviate, additional events (i.e., $\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_p$, where p is the number of measurements) may be produced, referring to deviations in these measurements. Once a fault is detected, the observer stops tracking the state trajectory, otherwise the faulty behaviors will be compensated for, and symbol generation will be incorrect. Future estimated values provided for symbol generation are computed using only the model of the system.

In order to represent the dynamics of the deviation, two features are extracted from the change in a residual, and these are abstracted symbolically as `+`, `0`, and `-` symbols, representing above, at, and below nominal, respectively. The two features capture (i) whether or not a discontinuity occurred, and its sign, and (ii) the first-order change of the residual, as illustrated in Fig. 11 for deviations in the positive direction. For example, a positive discontinuity, or jump, with a negative first-order

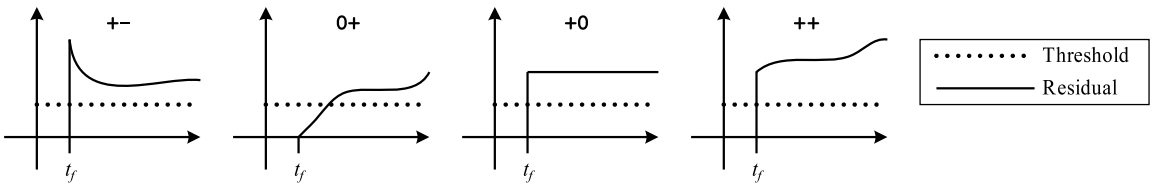


Figure 11: Symbolic abstractions of measurement deviations.

Table 2: Qualitative Symbols

Symbol	Description
`0`	No change in the residual (or its slope)
`+`	Increase in the residual (or its slope)
`-`	Decrease in the residual (or its slope)
`X`	Estimated and measured values are both zero or nonzero
`N`	Estimated value is zero and measured value is nonzero
`Z`	Estimated value is nonzero and measured value is zero

change is abstracted to `+-`, and no discontinuity with a positive first-order change is abstracted to `0+`. It is assumed that discontinuities will be detected at the point of fault occurrence, since discontinuities can be immediately differentiated from the system noise.

Assumption 6 (Discontinuity Detection)**.** Discontinuities are detected at the point of fault occurrence.

In addition to the increasing/decreasing values, we extend symbol generation by introducing specific symbols that are helpful in distinguishing parametric from discrete faults. Because discrete faults cause mode changes, they effectively cause some variables to change from nonzero to zero or change from zero to nonzero. For example, when a switch opens, the current through it becomes zero. Parametric faults, however, cannot cause this behavior with a finite change in magnitude. This behavior forms a special class of discontinuities, and we refer to it as discrete change behavior. If the variables that may exhibit the discrete change behavior are measured, then knowing whether a variable exhibited a discrete change or not provides additional discriminatory information. Therefore, we include an additional symbol denoting the discrete change in the measured value from the estimated value calculated by the hybrid observer. The symbol `N` denotes zero to nonzero behavior, the symbol `Z` denotes nonzero to zero behavior, and the symbol `X` denotes no observed discrete change.

Fig. 12 illustrates the alternatives for the discrete change feature. The symbol `Z` may be generated if a switching element unexpectedly turns off or if a switching element becomes stuck in the off state when it is expected to turn on. The symbol `N` may be generated if a switching element unexpectedly

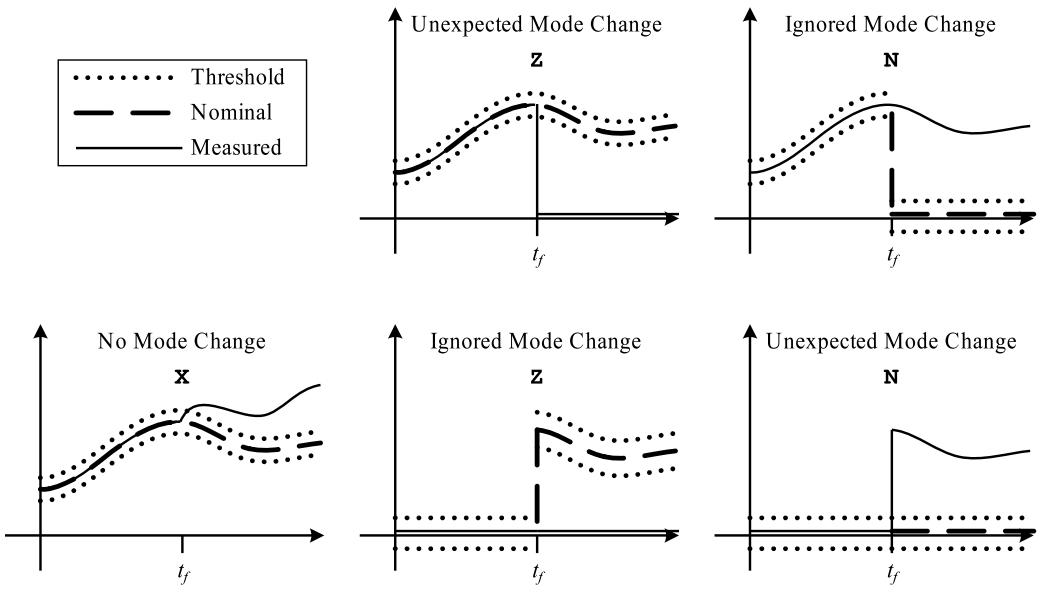


Figure 12: Symbol generation for the discrete change feature.

turns on if a switching element becomes stuck in the on state when expected to turn off. Note that the symbols are generated with respect to the expected nominal behavior.

Table 2 summarizes the computed features. The symbolic features are computed robustly using the Z-test on the residual. The initial change in a measurement is provided at the time of fault detection for the measurement. If the residual was positive, we take a `+`, and if negative, a `-`. After a deviation is detected in a measurement, then the slope of the residual is calculated. Again using the Z-test, an initial value is computed over a small window starting from the point of fault detection, from which the slope of the residual is determined over another small, but larger window (e.g., 15 samples) after the end of the smaller window [44]. If the slope is determined to be `+` or `-` within that window, it is reported. Otherwise, it is reported as `0`, implying that the true slope is either zero or unknown but very small. Based on the initial direction of change and the computed slope, we determine whether a discontinuity has occurred. If the initial change matches the computed slope, we do not classify this as a discontinuity. So, for example, a `+` in the magnitude and a `+` in the slope will be reported as `0+`, whereas a `+` in the magnitude and a `-` (or `0`) in the slope will be reported as `+-` (or `+0`). This strategy assumes that discontinuities of the form `++` or `--` will not occur, which is a practical assumption for parametric faults, because for parametric faults these symbols imply an unstable system. These symbols may occur for sensor faults, however, for example, some sensor faults may cause an immediate increase followed by a smooth increase, i.e., a `++`. The symbol generator cannot determine these types of discontinuities, therefore, it will only report elements

of the set $\{\texttt{+-}, \texttt{-+}, \texttt{0+}, \texttt{0-}, \texttt{+0}, \texttt{-0}\}$. A more advanced symbol generation technique is necessary to correctly identify `++` and `--`.

To compute the discrete change feature, we do not use the residual, but use the observed and estimated values. After a fault is detected, we compute the mean of the observation $y(t)$ and the mean of the estimate $\hat{y}(t)$ over a small window, W_3, to compute the discrete change feature:

$$\mu_y(t_d) = \frac{1}{W_3} \sum_{i=t_d}^{t_d+W_3-1} y(i) \tag{6}$$

$$\mu_{\hat{y}}(t_d) = \frac{1}{W_3} \sum_{i=t_d}^{t_d+W_3-1} \hat{y}(i), \tag{7}$$

where t_d is the time of fault detection. We wish to determine whether each signal belongs to a population with zero mean, and choose the variance of the population to be the variance of the residual defined by $y(t) - \hat{y}(t)$, $\sigma_r^2(t)$, as a good approximation of the true variance of the zero-mean distribution we wish to match to. So, the thresholds ε_{dc}^- and ε_{dc}^+ are computed as:

$$\varepsilon_{y_{dc}}^+ = \varepsilon_{\hat{y}_{dc}}^+ = z^+ \frac{\sigma_r(t_d)}{\sqrt{W_3}} + E_{dc} \tag{8}$$

$$\varepsilon_{y_{dc}}^- = \varepsilon_{\hat{y}_{dc}}^- = z^- \frac{\sigma_r(t_d)}{\sqrt{W_3}} + E_{dc}, \tag{9}$$

where E_{dc} is a modeling error term. These thresholds are the same for fault detection (Equations 4 and 5), only they are computed for $y(t)$ and $\hat{y}(t)$ rather than $r(t)$. If $\mu_y(t_d)$ is outside its bounds, we say it is nonzero, otherwise we say it is zero. Similarly, if $\mu_{\hat{y}}(t_d)$ is outside its bounds, we say it is nonzero, otherwise we say it is zero. If the estimate is nonzero and the measurement is zero, we report `Z`, and if the estimate is zero and the measurement is nonzero, we report `N`, otherwise, we report `X`.

Summary

We extend the TRANSCEND and Hybrid TRANSCEND methodologies for multiple fault diagnosis in hybrid systems. To keep the diagnosis efficient, we incorporate a candidate cardinality limit and exploit the concept of minimality. The proposed approach handles persistent and abrupt faults that may be parametric or discrete. Like Hybrid TRANSCEND, we also restrict the possible mode trajectories after fault occurrence by assuming the hybrid observer has executed a limited number of mode changes between the actual time of fault occurrence and the time of fault detection.

We extend Hybrid TRANSCEND to an event-based architecture. We perform system tracking of hybrid behaviors using a hybrid observer. Fault detection and symbol generation are defined as tests of statistical significance to avoid false alarms. In contrast to many approaches in continuous and hybrid systems diagnosis, diagnostic reasoning operates using computed symbolic features, rather than operating based purely on mode estimation, which is computationally more efficient.

CHAPTER IV

MODELING FOR DIAGNOSIS

Model-based diagnosis requires rich models of system behavior, where both the nominal and faulty behaviors of the system need to be correctly captured in a concise manner. Models for diagnosis must meet the following requirements.

1. **Level of Abstraction.** Diagnosis models should only represent the behaviors needed for diagnosis. Because diagnosis requires reasoning over the system model, the representation of the system must model the behaviors at the right level of abstraction to support efficient reasoning procedures. Diagnostic reasoning often uses cause and effect knowledge of the system [1,3,20,59,89]. Therefore, diagnostic models should include both causal and, for dynamic systems, temporal information, so that the possible causes of deviations from nominal behavior can be identified, and the future effects of possible candidates can be predicted. It is often useful to have different models at different levels of abstraction for different parts of the diagnosis problem.

2. **Executable Models.** For dynamic systems, the system behavior changes with time, so it must be tracked in order to provide a correct reference for deviations from nominal behavior. Therefore, the system model should be capable of efficiently simulating the system for generating correct behavior over time. The model should contain enough information so that observed faulty behavior can be mapped back to faulty components.

3. **Component-based Models.** The modeling paradigm should support component-based models, which enables reuse of component models, and thus decreases the modeling effort. Further, this provides a framework for model-based component fault isolation. Components should be defined in a common framework with common sets of interfaces between them, so that their interactions are clear and precise, and composition principles can be applied to derive system behavior from the component models.

Bond graphs define a language for energy-based modeling of physical systems [113]. They retain system topology and support first principles, physical systems modeling. Further, they allow for automatic generation of system equations [113]. They are particularly suitable for diagnosis because they incorporate causal and temporal information required for deriving and analyzing fault

transients [20, 28]. Furthermore, components can be explicitly represented as bond graph elements, which makes it easier to link observed fault transients to parameter value changes in the system components [20, 45]. Due to their advantages, several model-based diagnosis methods have been developed using the bond graph paradigm [14, 21, 114, 115].

Hybrid bond graphs (HBGs) extend bond graphs by allowing ideal binary switching of component connections. Because HBGs are based on bond graphs, they support automatic generation of system equations, simulation models [108, 109, 116], and observers [14, 22, 50]. HBGs also allow for component-oriented modeling [117].

Hybrid TRANSCEND uses the HBG modeling paradigm for modeling of hybrid systems for fault diagnosis. This dissertation also uses HBGs, and extends the HBG language to model sensors, sensor faults, and discrete faults. Preliminary results have appeared in [36,37]. The modeling paradigm has been used for domain-specific contributions for the robotics and electrical power systems domains, provided in the case studies found in Chapter VIII and also [28,95], and in Chapter IX and also [36, 37, 86, 116], respectively.

The diagnosis model in TRANSCEND, the temporal causal graph, is generated automatically from the bond graph model, and is used for to identify possible causes of faulty behavior, and to predict future behavior of faults [20]. We extend the temporal causal graph and its derivation procedure for the extensions made to the HBG modeling language, so that it can be used for isolation of both parametric and discrete faults.

We begin by describing the bond graph modeling language, followed by a description of the hybrid bond graphs and our extensions. We then discuss fault modeling, after which we detail the diagnosis model and how to it is generated automatically from a HBG model, given the extensions made to HBGs.

Bond Graphs

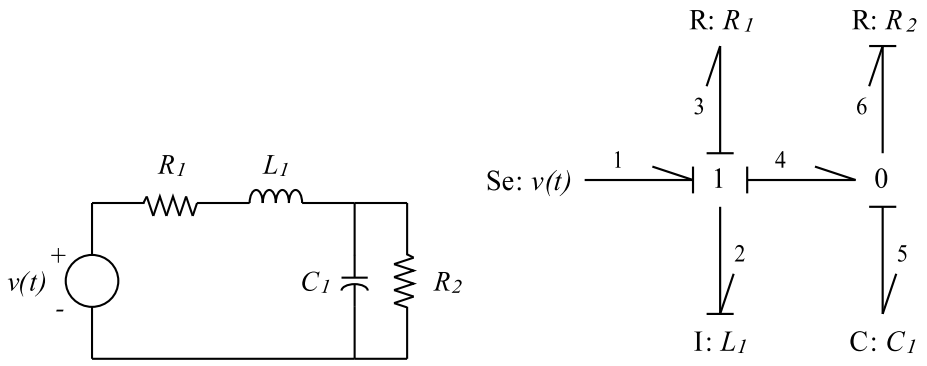
Bond graphs are directed graphs defined by a set of vertices and a set of edges. In bond graphs, vertices represent components. The edges, called *bonds* (drawn as half arrows), represent ideal energy connections between the components. Associated with each bond are two variables: *effort* and *flow*, denoted by e_i and f_i, respectively, where i denotes the bond number. The product $e_i \times f_i$ defines the rate of energy transfer through the bond. In the electrical domain, these variables map to voltage and current, whereas in the mechanical domain, they map to force and velocity, respectively.

Bond graph elements interface with bonds through *ports*. Bond graphs contain one-port, two-port, and multi-port elements. One-port elements model energy dissipation as resistances (R: R, where $e = Rf$), energy storage as capacitances (C: C, where $\dot{e} = \frac{1}{C}f$) and inertias (I: I, where $\dot{f} = \frac{1}{I}e$), and energy sources as sources of flow (Sf: u, where $f = u$) and effort (Se: u, where $e = u$). Two-port elements for energy transformation include transformers (TF: n, where $e_1 = ne_2$ and $f_2 = nf_1$) and gyrators (GY: r, where $e_1 = rf_2$ and $e_2 = rf_1$). One- and two-port elements connect to junctions, which are the multi-port elements. 1-junctions represent points of common flow (e.g., series connections), where all f are equal and $\sum e = 0$. The directions of the bonds determine the signs of the efforts in the summation. 0-junctions represent points of common effort (e.g., parallel connections), where all e are equal and $\sum f = 0$. The directions of the bonds determine the signs of the flows in the summation. Therefore, junctions represent idealized energy exchange ports, where power and energy are conserved. The constituent equations of the bond graph elements form a set of differential algebraic equations that describe the continuous behavior of the system.

Signal links (drawn as full arrows) are also used in the bond graph language, and represent information transfer pathways. Each link is associated with a single effort or flow variable. Nonlinearities can be introduced via *modulated elements*, which express nonlinear component parameters as algebraic functions of endogenous variables and exogenous inputs. Modulated elements are denoted by an M prefixed to the standard bond graph element name. The inputs to modulated elements are connected using signal links. For example, it is typical in battery models that the resistance is a function of the amount of charge left in the battery. The corresponding resistance, represented as an MR in the bond graph model, would express that function.

Example. Fig. 13 illustrates a system from the electrical domain. The circuit, shown in Fig. 13a, consists of a voltage source Se with input $v(t)$, resistors R_1 and R_2, inductor L_1, and capacitor C_1. Its bond graph is given in Fig. 13b. The current (i.e., flow) through the voltage source, resistance R_1, inductance L_1, and the rest of the circuit is equal, therefore, these elements are connected through a 1-junction. Elements C_1 and R_2 are in parallel so have the same voltage (i.e., effort), and, therefore, are connected through a 0-junction.

Sensors can also be modeled in bond graphs. If the physical processes of the sensor needs to be modeled, then the sensor can be instantiated using bond graph elements. Typically, it is sufficient to represent the output equation of the sensor. For example, if the voltage (effort) over C_1, $e_5(t)$, is measured, then the sensor output, $y(t)$, can simply be modeled as $y(t) = e_5(t)$.



(a) Schematic. (b) Bond graph.

Figure 13: Circuit example.

Causality

A bond graph captures the dynamics of a continuous system through a set of constituent equations. These equations form a set of differential algebraic equations, but often, an explicit formulation, such as state-space equations, is preferred, e.g., for use in deriving observers such as Kalman filters. *Causality* in bond graphs assigns computational directions to the bond variables, which is useful in deriving computational forms [113]. The causality of a bond is depicted using a causal stroke on one end. For example, in Fig. 13b, the causal stroke on bond 3 implies that it imposes flow on R_1, which responds with an effort that is imposed on the 1-junction.

Energy storage elements can be placed in either integral or derivative causality. In integral causality, the constituent equations are expressed in their integral form. For example, the equation for a capacitor in integral causality is $e = \int_0^t \frac{1}{C} f dt$. The effort, e, is computed by integrating the flow, f. In derivative causality, the constituent equations are expressed in their derivative form. For example, the equation for a capacitor in derivative causality is $f = C\frac{de}{dt}$. The computational direction is different here, because the flow, f, is computed using the derivative of the effort, e. Derivative causality leads to issues with simulation, therefore, integral causality is the preferred causality [113]. Throughout this dissertation, we assume the energy storage elements operate in integral causality.

Assumption 7 (Integral Causality)**.** All energy storage elements of a bond graph model operate in integral causality.

Each bond graph element has a set of causal constraints, shown in Fig. 14 with the constituent equations in their explicit form. A source of effort imposes effort, therefore its bond must impose effort on the connected junction. With sources of flow, the bond imposes flow on the adjacent

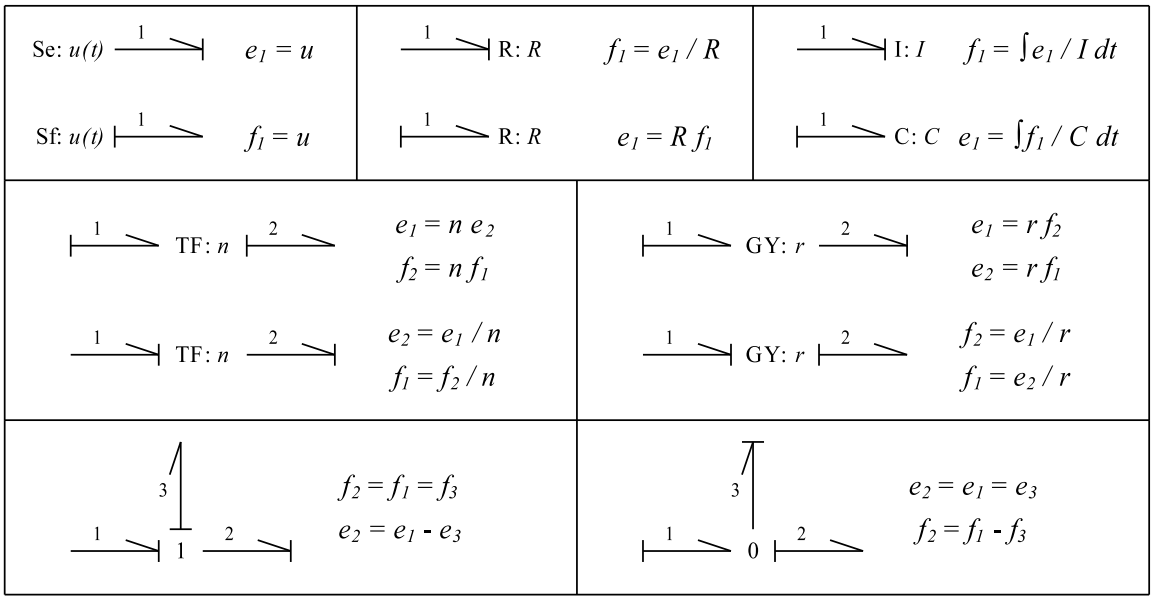


Figure 14: Causal constraints and explicit equation forms of bond graph elements.

junction. The equations for resistances are algebraic, therefore, their bond has no causal constraints. Under integral causality, bonds attached to capacitors must impose flow on the capacitor, and bonds attached to inertias must impose effort on the inertia. The equations for transformers and gyrators are algebraic as well, so can be in two different causal forms.

Associated with each junction, we define a special bond known as a *determining bond*. The determining bond for a 1-junction is the unique bond that determines its flow value. The effort value of this bond is an algebraic function of the effort values on all all other bonds incident on this junction. Similarly, the determining bond for a 0-junction is the unique bond that determines its efforts, and all other bonds determine the flow on that bond by an algebraic function of their flow values.

Definition 6 (Determining Bond)**.** The *determining bond* for a 1-junction (0-junction) is the unique bond that determines the flow (effort) for that junction.

Based on the set of constraints, causality can be assigned systematically using the *Sequential Causal Assignment Procedure* (`SCAP`) [113]. For example, consider the circuit of Fig. 13b. Because of the Se, bond 1 must impose effort on the 1-junction. Under integral causality, bond 2 imposes flow on the 1-junction, becoming its determining bond. So, bonds 3 and 4 (and 1) must impose effort on the 1-junction. Under integral causality, bond 5 imposes effort on the 0-junction, so bond 6 (and 4) must impose flow on the 0-junction, and all bonds have causality assigned. Causality cannot always be determined, and this usually indicates an incorrect model [113]. Therefore, we assume

that bond graph models have valid causality assignments. Given a causal bond graph and using the equation forms of Fig. 14, it is straightforward to generate a computational form, e.g., state-space equations, signal flow graphs, or block diagrams (e.g., [108, 109, 113, 116]).

Hybrid Bond Graphs

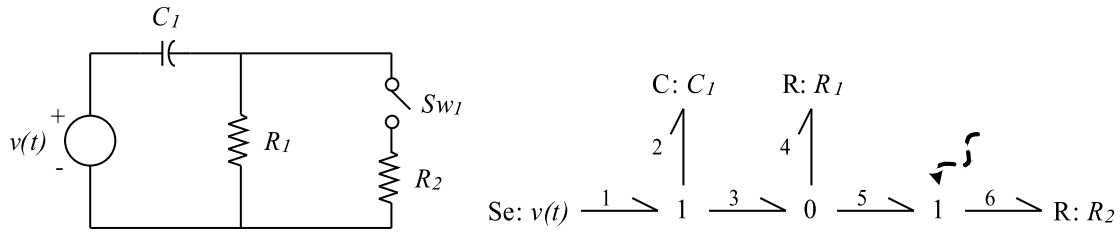
Bond graphs define a modeling language for continuous systems. Hybrid bond graphs (HBGs) extend bond graphs to a hybrid modeling framework, in a way that they produce physically verifiable hybrid models of system behavior [48]. Unlike other approaches that use switching bonds, switch elements, or modulated transformers and gyrators to model switching in bond graphs [118–121], HBGs introduce *controlled junctions*. Controlled junctions act as ideal switches, enabling a junction to be in either the on or the off state. When a 1-junction is off, it behaves as a source of zero flow, so it imposes $f = 0$ on all its bonds. Similarly, when a 0-junction is off, it acts as a source of zero effort, and imposes $e = 0$ on all its bonds. When on, controlled junctions behave as normal junctions.

The switching behavior of a controlled junction is defined by a *control specification* (CSPEC), modeled as a finite automaton [14, 48]. A CSPEC defines a finite number of states. The state transitions may be attributed to controlled or autonomous events. Associated with each state is an output, which determines whether the junction is on or off. The traditional CSPEC definition, which we will later extend, is as follows.

Definition 7 (Control Specification)**.** A *control specification* is a tuple $\mathcal{M} = (S, E, \delta, o, s_0)$, where S is the finite set of states, E is the set of events, $\delta : S \times E \rightarrow S$ is the transition function, $o : S \rightarrow \{on, off\}$ is the output function, and $s_0 \in S$ is the initial state.

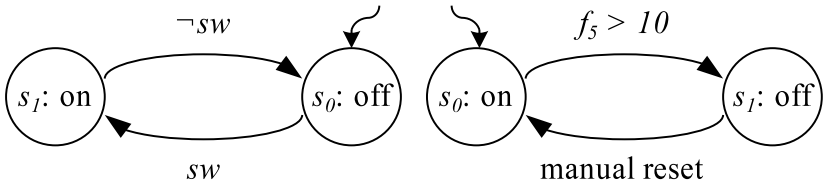
Example. Fig. 15 shows a switched circuit. Essentially, the switch, Sw_1, functions as a series connection that turns on and off. So, in the HBG, the switch is modeled as a controlled 1-junction. In Fig. 15b, the controlled junction modeling the switch is denoted with the dashed arrow. When the junction is on, the switch is closed, and when the junction is off, the switch is open, and the current through R_2 is forced to zero.

CSPECs for the circuit switch are given in Fig. 16. The states are labeled with the state name and the state output. The CSPEC for a relay controlled by switching signal, sw, is shown in Fig. 16a. When sw is true, an event is generated and the junction moves to s_1, where the junction is on. When sw is false, it moves to s_0, where the junction is off. For a circuit breaker (Fig. 16b), the junction behavior is autonomous. It is initially in s_0, where the junction is on. When the current through



(a) Schematic. (b) Bond graph.

Figure 15: Switched circuit example.



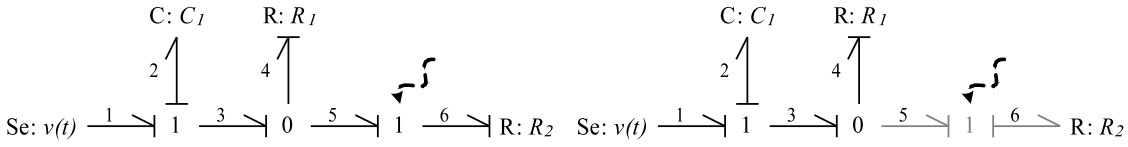
(a) Relay CSPEC. (b) Circuit breaker CSPEC.

Figure 16: Circuit switch CSPECs.

the circuit breaker, f_5, exceeds the value 10, a corresponding event is generated and the CSPEC moves to s_1, where the junction is off.

Because HBGs define switching locally, they provide a concise representation of the hybrid model. This contrasts to hybrid automata models [98], where system modes are pre-enumerated, which, in practical systems, is not often feasible. In HBGs, the system mode is defined implicitly by the individual states of all the CSPECs. Because a single mode change may correspond to multiple junctions switching their mode, events may be shared over different CSPECs. Given an event e and the current system mode q, the new system mode, q', is given by $q' = \mu(e, q)$, where the system mode transition function, μ, simply applies e to all CSPECs, and obtains the new CSPEC states for each controlled junction. Associated with each mode q is a continuous bond graph, where each controlled junction's CSPEC defines whether the junction is on or off. The computational model for each mode (i.e., state-space equations, signal flow graphs, block diagrams, etc.) can be derived systematically using standard bond graph methods [113].

Because the system equations are different in each mode, causality assignments may change from mode to mode. Fig. 17b shows the HBG when the junction is off, with the off state denoted by a gray color of the junction and its bonds. The off junction imposes flow on all adjacent elements, because it acts as a source of zero flow. Fig. 17a shows the HBG when the junction is on. In this mode, the junction now imposes effort on R_2.



(a) Causal HBG for the on mode.

(b) Causal HBG for the off mode.

Figure 17: HBGs for the switched circuit.

The Hybrid Sequential Causal Assignment Procedure (Hybrid `SCAP`) is used to assign causality to HBGs [108,109,116]. Rather than simply applying `SCAP` globally every time the mode is changing to obtain the causality for the new mode, the main idea behind Hybrid `SCAP` is that, when a mode change occurs, only a small part of the model usually changes, so causality can be reassigned incrementally rather than globally. This approach enables the computational model for a new mode to be automatically generated from the previous mode efficiently [108, 109, 116].

Like bond graphs, HBGs support component-based modeling practices. A component-based HBG modeling language is described in [22,117]. System components are defined as HBG fragments that interface through energy ports (for bonds) and signal ports (for signals).

Modeling Faults

Since we assume faults are persistent (Assumption 4), we classify faults only along the dimension of their modeling representation. We define hybrid system faults as either (*i*) *parametric faults* or (*ii*) *discrete faults*. Parametric faults are associated with changes in parameter values, and are useful for modeling partial failures or degradations in system components [14, 20, 99]. For example, a change in the electrical resistance can be modeled by a change in the resistance parameter value. Discrete faults, which may be modeled as fault events or fault modes, are associated with mode changes, and are useful for modeling switching failures in components [19, 35–37, 99]. For example, a relay may turn off by itself, or become stuck in a particular state. Like in DES approaches, we model all faults as unobservable events.

It should be noted that faults modeled using parameter changes can also be modeled as new modes if the fault magnitude is known, and this is the approach taken in discrete fault approaches for hybrid systems [15,18,30–32,75–77]. In most cases, fault magnitudes are unknown, and to cover a large number of possible fault magnitudes, many new modes are needed. Discrete faults can also be captured at a very detailed level as parameter value changes, for example, a switch turning off can be modeled as a resistance for the switch going to infinity, but this produces stiff, nonlinear

models that are difficult to simulate and analyze efficiently, and may introduce algebraic loops into the system equations, which makes analysis even more challenging.

Parametric Faults

Formally, we define a *parametric fault* as follows.

Definition 8 (Parametric Fault)**.** A *parametric fault* is an unexpected change in the value of a system parameter in the model.

Parametric faults may be *additive* or *multiplicative* in nature [10,52]. Additive faults are modeled as unknown inputs to the system that do not affect the evolution of the system state. For example, consider the sensor model $y(t) = y'(t) + b(t)$, where $b(t)$ is a bias term added to the true value, $y'(t)$, to produce the observed value, $y(t)$. Nominally, the bias is zero, but when a bias fault occurs, the bias is nonzero and affects the sensor output in an additive way.

Multiplicative faults are modeled as changes in internal system parameters, and they affect the system dynamics and, therefore, the evolution of the system state. For example, a resistance change in a circuit will affect the voltage and current through the resistor, which in turn will affect other voltages and currents throughout the circuit. Sensor faults may also be multiplicative if there is feedback from the sensor to the plant through a controller.

We represent parametric faults as unobservable events that change a parameter value. In HBGs, system components appear as HBG model parameters, so changes in these parameters model multiplicative faults. For example, a change in capacitance can be modeled as a change in the capacitance value of the corresponding C element in the HBG model. Additive faults map to source elements (Se and Sf) in the HBG. For example, a sensor bias can be parameterized as a source, where a change in the nominal zero value of the source represents a bias. In the circuit example of Fig. 15, parametric faults may include increase and decrease in resistance (R_1 or R_2) and capacitance (C_1) values. We denote parametric faults by the parameter name and its change, e.g., R_1^+ denotes a parametric fault that increases the value of R_1. Unlike discrete faults, parametric faults do not directly induce changes in system mode.

Parametric faults may also have different profiles, i.e., the time-varying behavior of the parameter value change may be arbitrary. In most cases, changes are abstracted as either *abrupt* or *incipient* [52]. Abrupt faults manifest at full magnitude immediately. Incipient faults represent slow degradations and other effects such as sensor drift, and are not considered here (Assumption 5).

Definition 9 (Abrupt Fault)**.** An abrupt parametric fault in parameter θ is a step change in the component parameter value, i.e., for $t \geq t_f$, $\theta(t) = \theta(t_f^-) + \Delta\theta(t)$, where $\Delta\theta(t) = A$.

The faulty parameter value, $\theta(t)$ for $t \geq t_f$, is represented as the sum of its nominal value, i.e., its value before the fault, $\theta(t_f^-)$, and the fault function, $\Delta\theta(t)$. The fault profile depends on the time-varying behavior of the fault function. For an abrupt fault, the fault function is simply a constant A, where the magnitude is represented by the value of A. Note that the definition implies that faults are persistent, i.e., the parameter is faulty for all $t \geq t_f$.

Discrete Faults

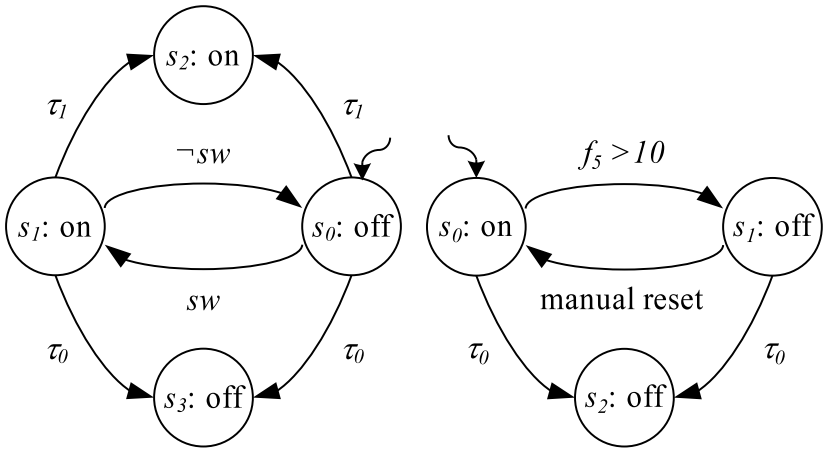
An important contribution of this dissertation is the modeling and diagnosis of both parametric and discrete faults within an integrated framework. Formally, we define a *discrete fault* as follows.

Definition 10 (Discrete Fault)**.** A *discrete fault* is a discrepancy between the actual and expected mode of a switching element in the model.

A discrete fault causes a switching element to enter a fault mode. Such a switching element may be part of an actuator, the plant, or a sensor. For example, a type of sensor failure where the sensor output is always zero can be modeled as a discrete fault. In the nominal mode, the sensor model is $y(t) = y'(t)$, but in the fault mode, the sensor model is $y(t) = 0$. Another fault mode may be one where the sensor becomes stuck at its value at the time of fault occurrence, i.e., the sensor model for the fault mode is $y(t) = y(t_f)$. Discrete faults in the circuit example include switch malfunctions. For example, the switch may be commanded to turn on, but remain stuck off. Also, it may unexpectedly turn on or off without a command.

We model a discrete fault as an unobservable fault event that brings the component to the corresponding fault mode. In HBGs, mode changes are modeled using controlled junctions whose modes are determined by CSPECs, so discrete faults are captured as unexpected changes in CSPEC state. We introduce new unobservable fault events in the CSPEC and link discrete faults to them. Mode changes in components may correspond to many junctions changing mode, so these fault events may be shared among the different CSPECs of the component. The linking of discrete faults to fault events in the CSPEC gives, as with parametric faults, a one-to-one mapping between model entities and faults. Thus, we provide an extended definition of the CSPEC to include discrete faults.

Definition 11 ((Extended) Control Specification)**.** A *control specification* is a tuple $\mathcal{M} = (S, E, \delta, o, s_0)$, where S is the finite set of states, $E = E_o \cup E_u$ is the set of observable and unobservable



(a) Relay CSPEC. (b) Circuit breaker CSPEC.

Figure 18: Circuit switch extended CSPECs.

(fault) events, $\delta : S \times E \rightarrow S$ is the transition function, $o : S \rightarrow \{on, off\}$ is the output function, and $s_0 \in S$ is the initial state.

Example. The extended CSPECs for the circuit of Fig. 15 are given in Fig. 18. For the relay CSPEC (Fig. 18a), we introduce fault events τ_0 and τ_1. When τ_1 occurs, the CSPEC moves to s_2, where the junction is stuck on. If the junction was previously off, then this fault manifests in the measurements immediately, i.e., the switch turns on by itself. Otherwise, it will only manifest when sw becomes false, i.e., the switch becomes stuck on. The case is similar for the τ_1 event. For the circuit breaker CSPEC (Fig. 18b), only the stuck off fault, τ_0, is appropriate, and the behavior is similar. The circuit breaker may open due to the current limit being exceeded, which is nominal behavior, or may turn off due to a fault, i.e., it turns off when the current limit has not yet been exceeded.

Because we consider faults to be persistent, all discrete faults are essentially "stuck" faults, i.e., a switching element enters a fault mode and persists in that mode. As illustrated by the example, discrete faults may manifest in the sensors in two ways. In one case, if the CSPEC transitions from a nominal state to a fault state, both of which map to the same output, then the discrete fault will only be detected when a mode change is expected to occur, but does not. For example, a relay is on, then becomes stuck on, and some time after, a command is issued to turn it off. In the other case, if the CSPEC transitions from a nominal state to a fault state, where the two states map to different outputs, then the discrete fault should be detected immediately. For example, a relay is on, and then becomes stuck off. In the absence of a controller command to turn off, there is an immediate discrepancy between the expected and actual modes of the relay.

Temporal Causal Graphs

HBG models capture the continuous and discrete dynamics of a system. Since they represent the system equations, they are useful for designing observers and simulating system behavior. However, we need a representation of the system behavior that explicitly incorporates cause and effect relationships between the faults and the measurements. The *temporal causal graph* (TCG) is an abstraction of the continuous behavior to a qualitative domain [20]. Using the TCG, we can reason efficiently about the qualitative behavior of the system when a fault occurs with respect to nominal behavior.

A key advantage for fault diagnosis of the BG and HBG modeling languages is the use of causality. Causality maintains the causal relations between system variables, so changes in measured variables can be mapped back to possible sources of change, and fault effects can be propagated forward to predict effects on the measurements. The TCG uses the causality of a bond graph model to capture this information by explicitly separating out the variables and signals from the bond graph and making the relations between variables clear. The TCG is essentially a signal flow graph with qualitative edge labels, and captures the propagation of qualitative fault effects on the measurements. We provide an extended form of the TCG to encode the effects of both parametric and discrete faults that is defined as follows.

Definition 12 (Temporal Causal Graph)**.** A temporal causal graph is a labeled directed graph $\mathcal{G} = (V, E, L, l_E)$ where $V = V_X \cup V_T$ is a set of vertices representing system variables X and failure events T, $E \subseteq V \times L \times V$ is a set of edges indicating causal and temporal relations between system variables, $L \subseteq \{=, +1, -1, N, Z\} \cup \{\theta : \theta \in \Theta\} \cup \{1/\theta : \theta \in \Theta\} \cup \{\theta\ dt : \theta \in \Theta\} \cup \{1/\theta\ dt : \theta \in \Theta\}$ is a set of labels (where Θ is the set of system parameters), and $l_E : E \to L$ is a labeling function that maps edges to labels.

Like a signal flow graph, the vertices of a TCG may represent system variables, such as the efforts and flows of a bond graph model. Additionally, vertices may represent fault events. The addition of vertices for fault events provides explicit sources in the TCG for discrete faults. Because the fault events are explicit in the TCG, the TCG can be used to analyze the effects of the occurrence of discrete faults.

The edges of the TCG are labeled according to the causal and temporal relations between system variables and fault events. Edges may connect two system variables, or connect a fault event to a system variable. For an edge (v_1, l, v_2) connecting two variables, the label l may be $=$, representing equality, $+1$, indicating direct proportionality, -1, indicating inverse proportionality, θ or $\frac{1}{\theta}$, indi-

cating a parametric relation (i.e., $v_2 = \theta v_1$ or $v_2 = \frac{1}{\theta} v_2$), or $\theta\ dt$ or $\frac{1}{\theta} dt$, indicating an integration (i.e., $v_2 = \theta \int v_1\ dt$ or $v_2 = \frac{1}{\theta} \int v_1\ dt$). Because the system parameters are explicit in the TCG, the TCG can be used to analyze the effects of changes in the values of the parameters, i.e., parametric faults. For an edge (v_1, l, v_2) connecting a fault event to a variable, the label l may be $+1$ or -1, but may also be N, indicating that the fault event will cause the connected system variable to go from zero to a nonzero value, or Z, indicating that the fault event will cause the connected system variable to go from a nonzero value to zero.

Constructing the TCG

For a particular mode, the TCG is constructed based on the system equations for that mode. Using bond graphs, this process is made easy. First, causality is assigned using `SCAP` [113], or, in the case of HBGs, may be reassigned based on the assignment of the previous mode using Hybrid `SCAP` [108, 109, 116]. Then, the bonds are converted to system variables (one effort and one flow per bond) and the bond graph elements are converted into labeled edges connecting the variables of their associated bonds (see Fig. 19). Signal links are converted into single edges, with the qualitative label corresponding to the qualitative relationship between the variables. This requires sensitivity analysis and may result in a label indicating an unknown relationship [20]. An algorithm for TCG generation is given in [22].

Example. Fig. 20 shows a bond graph and its corresponding TCG. The numbered bonds are converted to corresponding variables with the subscript indicating the bond number. For example, bond 3 becomes e_3 and f_3. The resistor R_1 relates these two variables, i.e., $e_3 = R_1 f_3$. The causality of the bond indicates that the 1-junction is imposing flow on R_1, and R_1 is imposing effort on the 1-junction. Therefore, the causal relationship is from f_3 to e_3. The label is R_1, which corresponds to the constituent equation of the resistor in the given causality. For the inductance, the relationship between f_2 and e_2 is an integration, hence the dt label. Junctions sum one type of variable according to the bond signs, and set the other type equal. For the 0-junction, bond 5 is determining the effort of the junction, so e_4 and e_6 are set equal to e_5. According to the bond signs, $f_4 = f_5 + f_6$. Since f_5 must be determined, f_4 and f_6 causally affect f_5 with labels $+1$ and -1, respectively.

We extend the TCG generation algorithm for Hybrid TRANSCEND, presented in [22], to perform two additional steps. First, we incorporate sensor models into the TCG in order to model sensor faults. We then incorporate discrete faults into the TCG. These are described next.

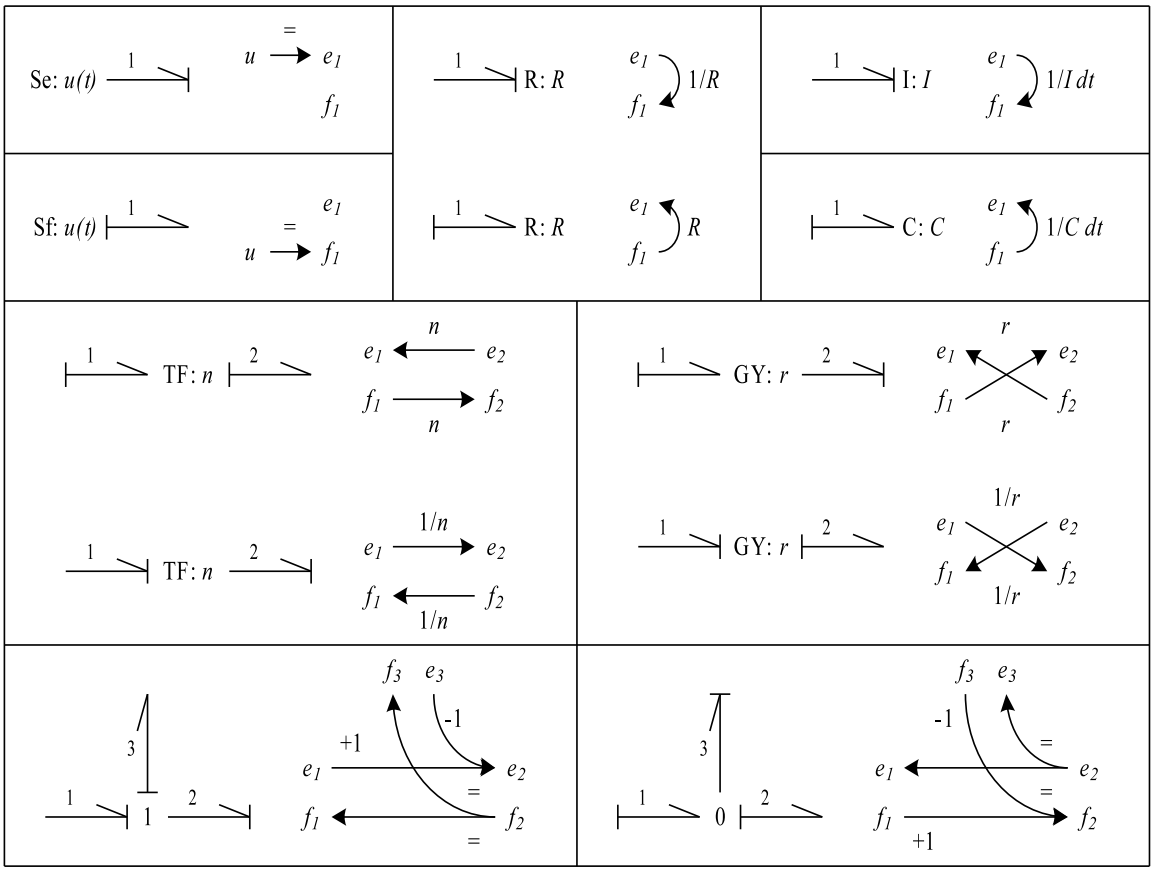
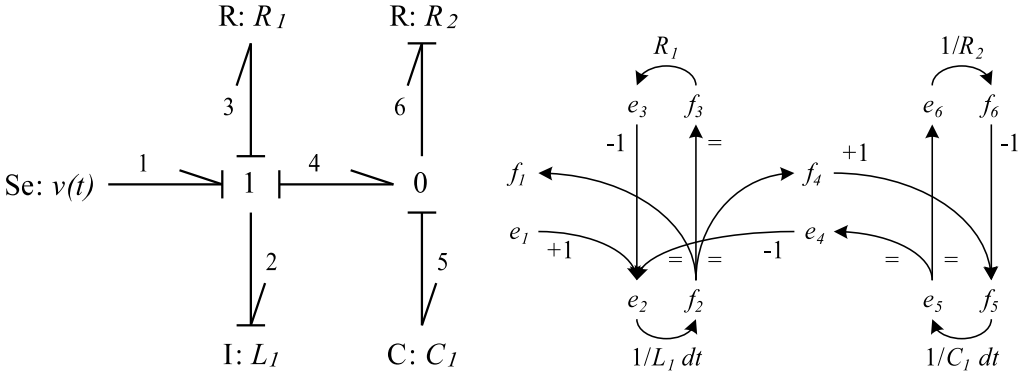


Figure 19: Temporal causal graph transformations.



(a) Bond graph.

(b) Temporal causal graph.

Figure 20: Temporal causal graph example.

Incorporating Sensors

If sensors are modeled using bond graph elements, their parameters appear explicitly in the TCG. Otherwise, they can be introduced by converting the sensor models into equivalent TCG fragments. For example, consider the sensor model $m_1(t) = f_6(t) + b_1(t)$, where $b_1(t)$ is a sensor bias term. The

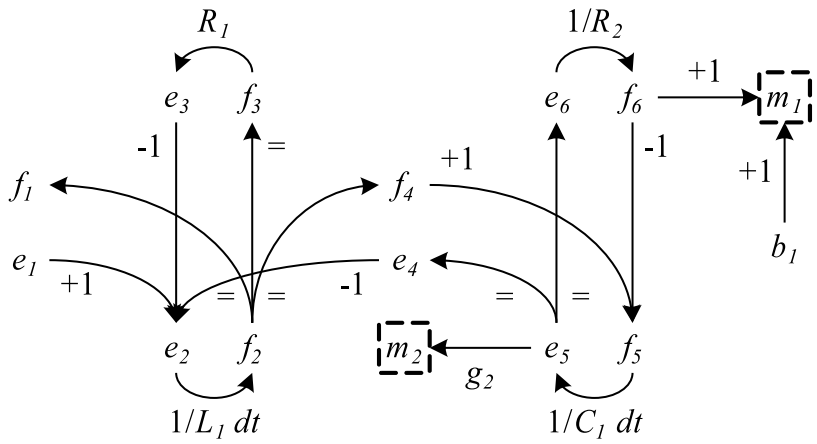


Figure 21: Temporal causal graph with sensors.

measured value of $f_6(t)$ will be the true value biased by $b_1(t)$. Fig. 21 illustrates how this is reflected in the TCG. Additional variables are added for the bias term $b(t)$ and the sensor measurement $m_1(t)$. To reflect the output equation, +1 edges are made from the bias variable to the sensor variable and the true flow variable $f_6(t)$ to the sensor variable. As another example, consider the sensor model $m_2(t) = g_2(t)e_5(t)$, where $g_2(t)$ is a gain term. The measured value of $e_5(t)$ will be the true value multiplied by the sensor gain $g_2(t)$. Fig. 21 illustrates this scenario as well. An additional variable for the sensor measurement, $m_2(t)$, is included, and an edge made from the true effort variable $e_5(t)$ to the sensor variable is made, labeled with the gain $g_2(t)$. More complicated sensor models (e.g., those including multiple fault terms or different fault profiles) may be modeled in this fashion as well.

Incorporating Discrete Faults

We also augment the TCG to capture the effect of discrete faults on the system variables by creating a new vertex in the TCG for each discrete fault event. Because discrete faults represent changes in CSPEC state and, therefore, junction mode, they must directly affect the variables associated with a junction. We create edges with $+1$, -1, N, and Z labels linked to the appropriate junction variables. We also introduce the junction mode variable p_i for each controlled junction i, which the discrete fault will also affect, allowing us to use sensors that measure the state of switching elements during fault isolation. The discrete fault event is added to the TCG and edges made to the junction variables if it is possible from the last known mode that the discrete fault could have occurred, given the logic of the junction's CSPEC.

In order to correctly connect the fault event to the junction variables, we must analyze what happens when a junction turns on or off. Consider first the case where a junction turns off. We need to determine which variables are immediately affected by the change in the junction mode. From bond graph causal analysis, each 1-junction (0-junction) that is on has a flow (effort) determining bond. When a junction turns off as the result of a fault, the determining flow (effort) will be immediately affected because it goes to zero. Take the case of a 1-junction. When it turns off, all the flows of its bonds are forced to zero. Therefore, the fault event causing the junction to turn off should immediately affect all the flow variables of the junction. Since there is a single flow that determines the other flows when the junction is on, this can be represented with an edge labeled with Z going from the fault event to the determining flow. Since the other flows are equal to this flow, the relationship is also captured on these variables by the single edge.

However, the flow of the switching 1-junction is not the only variable that is immediately affected. This is revealed by causal analysis. When a junction changes state in a hybrid bond graph, the causality of the bond graph changes. There are two ways in which the causal change can be absorbed.

1. The first way is through a resistor. The bond of a resistor can be in either causal form. If the change in causality of the determining bond propagates to a resistor, the resistor can absorb the change by switching causality.

2. The second way is though another junction of the opposite type simultaneously changing state. For example, if a 1-junction is adjacent to a 0-junction such that the determining bond of the 1-junction is the one connected to the 0-junction, and both junctions switch off simultaneously, then the causality of the connecting bond switches causality and absorbs the causal change.

Consider the case of a switching 1-junction. In the first scenario, since a resistor absorbs the causal change, the effort and flow of the determining bond are algebraically related. Therefore, if the flow is immediately affected, then the effort will be also, so an edge from the fault event to this variable is necessary. This is shown in Fig. 22 , assuming that the flow is positive (a gray coloring indicates the junction is off). If the effort and flow are related by a gain, then they will both go to zero, so the edge is labeled by a Z. This is shown in Fig. 23. If not, then they will change in the same direction if the algebraic path between the variables has an overall $+1$ label, and in opposite directions if the path has an overall -1 label. In physical systems, resistance should be positive, so the path should always have a $+1$ label and we can assume that the effort and flow of the determining bond change in the same direction.

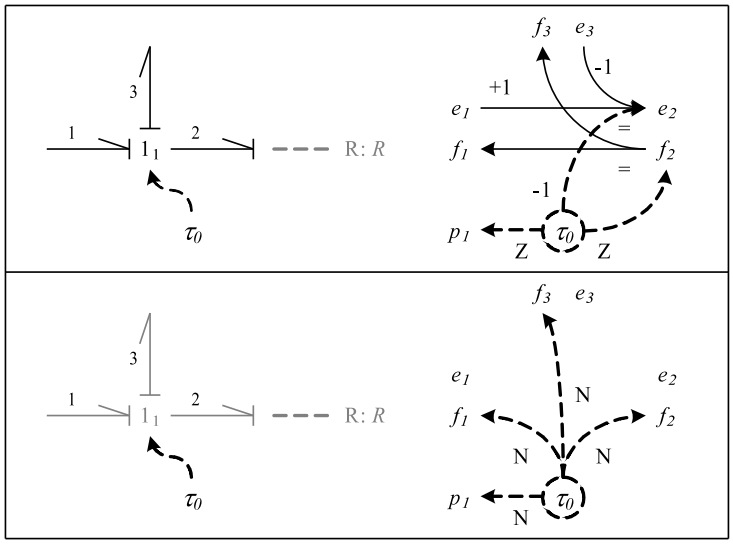


Figure 22: Temporal causal graph transformation for a switching 1-junction with a discrete fault where a non-adjacent resistor absorbs the causal change.

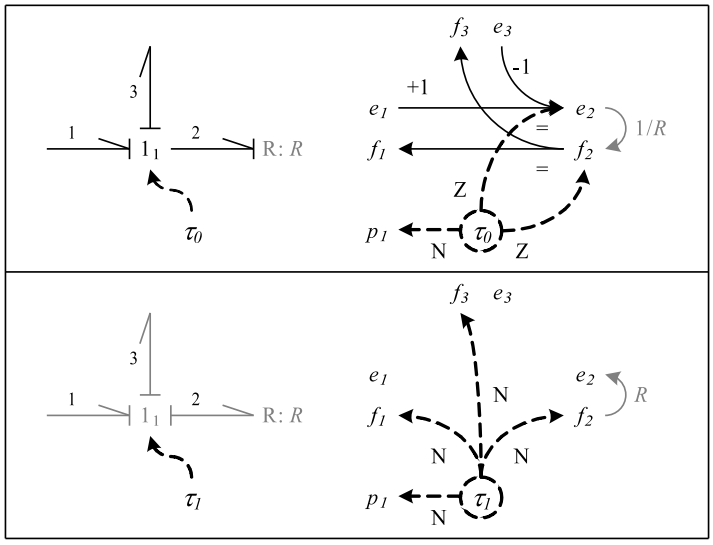


Figure 23: Temporal causal graph transformation for a switching 1-junction with a discrete fault where an adjacent resistor absorbs the causal change.

Assumption 8 (Consistent Direction of Change)**.** When a junction turns off, the effort and flow of its determining bond change in the same direction.

In the second scenario, the effort of the 1-junction's determining bond is determined by the 0-junction when both junctions turn off, and similarly the flow of the 0-junction's determining bond is determined by the 1-junction when both junctions turn off. Therefore those variables will be immediately affected, and go to zero, so edges are needed here as well and will be labeled by Z.

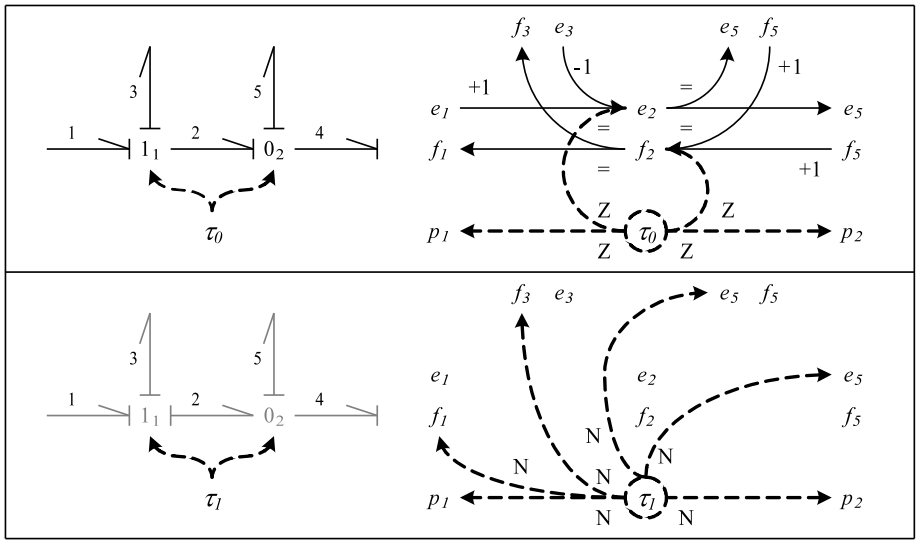
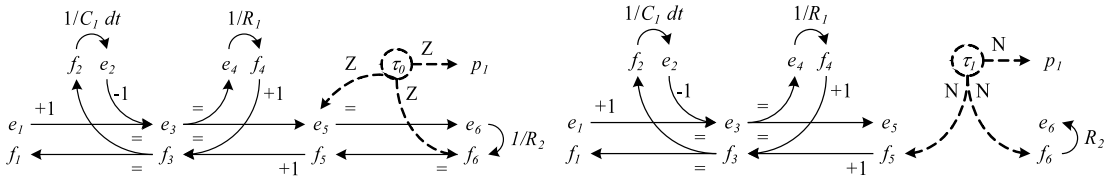


Figure 24: Temporal causal graph transformation for simultaneously switching adjacent 1- and 0-junctions with a discrete fault.



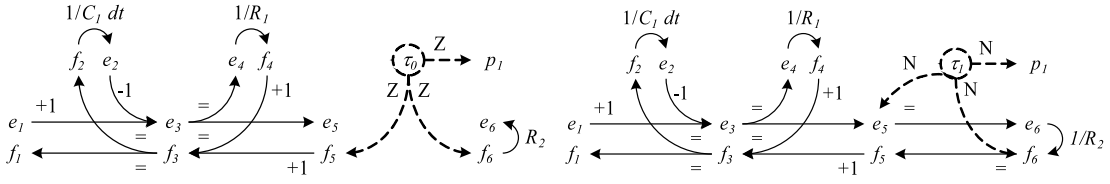
(a) TCG for the q_1 mode.

(b) TCG for the q_0 mode.

Figure 25: TCGs for the nominal modes of switched circuit.

This is shown in Fig. 24. When the junctions are expected to be on, the τ_0 fault that causes them to turn off directly affects the shared determining bond, bond 2, and so affects e_2 and f_2.

Example. Consider the circuit and the CSPECS given in Figs. 13 and 18, respectively. Recall that there is a τ_0 event causing the junction to turn off, and a τ_1 event causing it to turn on. Consider the mode where the relay is closed, i.e., the junction is on. The extended TCG for this mode is shown in Fig. 25a. In this case, R_2 absorbs the causal change when the 1-junction turns off. The on switch creates a configuration where a voltage is imposed on R_2, determines its current. An off switch, however, imposes zero current on R_2, and, therefore, determining its voltage. As a result, the τ_0 fault will immediately affect e_6, f_6, and p_1. Because e_5 is equal to e_6, we place the (τ_0, Z, e_6) edge as (τ_0, Z, e_5) instead, so that the causal order of affected variables is maintained. Note that in this case, a Z label can be used on the edge to e_5 because it is equal to f_6 times a gain, and since f_6 goes to zero, so will e_6 and e_5.



(a) TCG for the q_{τ_0} fault-induced mode.

(b) TCG for the q_{τ_1} fault-induced mode.

Figure 26: TCGs for the fault-induced modes of the switched circuit.

When a junction is off, then the analysis is easier. Consider the case of a 1-junction. When off, all its flows are zero and its bonds impose flow on all adjacent components. We need to represent the effects of the fault in this mode. When the junction turns on, all its flows will become nonzero, therefore, they are all immediately affected. So, we create edges labeled with N from the fault event to each flow. The TCG for the circuit example in this case is shown in Fig. 25b. The τ_1 fault will immediately affect f_5, f_6, and p_1.

Because discrete faults induce mode changes, it is also necessary to derive the TCG for the induced fault modes, especially since multiple junction mode changes may correspond to a single discrete fault. TCGs for the fault-induced modes are constructed the same way as the nominal modes, only the junction modes are changed to reflect the occurrence of the fault. The TCGs for the fault-induced modes for τ_0 and τ_1 are shown in Fig. 26. For example, in Fig. 26a, τ_0 is linked to an off junction, whereas in Fig. 25a, it is linked to an on junction.

Summary

Bond graphs provide a useful framework for physical systems modeling. Hybrid bond graphs extend the bond graph language to modeling hybrid systems. The model in each mode is a continuous bond graph, and the bond graph models for a new mode are derived from the previous bond graph model by turning junctions on and off. HBGs provide many advantages, such as compactly representing the hybrid behavior and incorporating causal information, so we use them to model hybrid systems and develop hybrid observers. Since faults are explicitly modeled in the HBG, the temporal causal graph, derived automatically from the HBG, can be used to relate observed discrepancies in the measurements to possible faults, and relate possible faults to predicted discrepancies. We extended the HBG language to include the modeling of sensors, sensor faults, and discrete faults. The temporal causal graph, which serves as the diagnosis model, was similarly extended to incorporate the modeling extensions, validated by causal analysis. Our methods are not entirely dependent on HBGs, but HBGs provide a systematic way to produce temporal causal graphs. The TCG is used

in the qualitative fault isolation algorithms that perform the hypothesis generation, prediction, and tracking steps. These methods are described in the two following chapters.

CHAPTER V

QUALITATIVE FAULT ISOLATION IN CONTINUOUS SYSTEMS

The qualitative model-based approach to fault isolation analyzes observed deviations from nominal behavior in the system measurements. These deviations are abstracted to qualitative increasing/decreasing values for magnitude and slope of the residuals, and are represented as events. This chapter builds up the theory for fault isolation in continuous systems based on measurement deviation events, and describes the associated algorithms for fault isolation. The results of this chapter extend the TRANSCEND methodology presented in [20, 45].

The basic diagnoser architecture is shown in Fig. 27. Given one or more initial measurement deviation events, σ_0, the diagnoser produces an initial diagnosis, i.e, a set of fault hypotheses, d_0. The initial diagnosis triggers the prediction scheme, which generates the set of predictions P_0 as fault signatures and relative measurement orderings for the hypothesized candidates. A tracking procedure refines candidates as more measurement deviation events are generated, and may generate new predictions P_i when new fault hypotheses are considered. After each new event, candidate tracking produces an intermediate diagnosis d_i, which represents a refined set of fault hypotheses. A user-specified stopping criterion terminates the fault isolation process. This criterion is based on isolation of a unique candidate, the number of measurements that have deviated, or a specified time limit.

The previously developed TRANSCEND algorithms apply to only single, abrupt, parametric faults. Further, the temporal order of the measurement deviations was not utilized to refine the fault candidates. One of the primary contributions of this dissertation research is the use of *relative measurement orderings*, which define predicted partial orders of measurement deviations for each

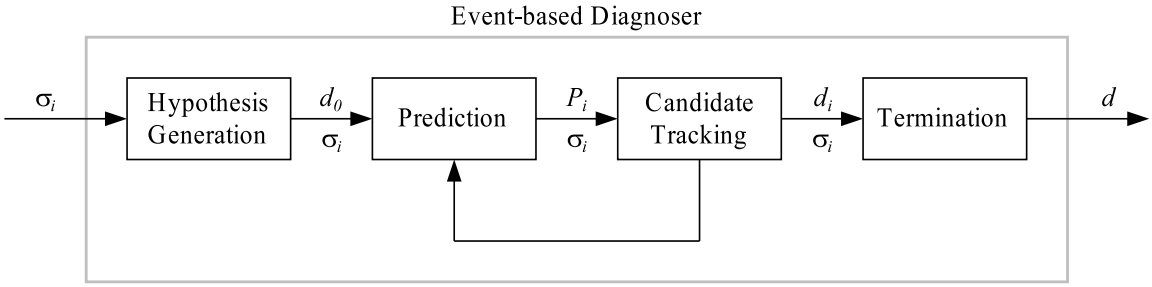


Figure 27: Event-based diagnoser architecture.

fault [27,28,95]. We show how the extended form of the temporal causal graph can be used to predict measurement orderings by extending the original prediction algorithm. Further, we relax the single fault assumption and and extend the TRANSCEND approach to include multiple fault isolation in continuous systems [38, 39]. The results provided in this chapter provide the framework for the diagnoser design for continuous systems, described in Chapter VII.

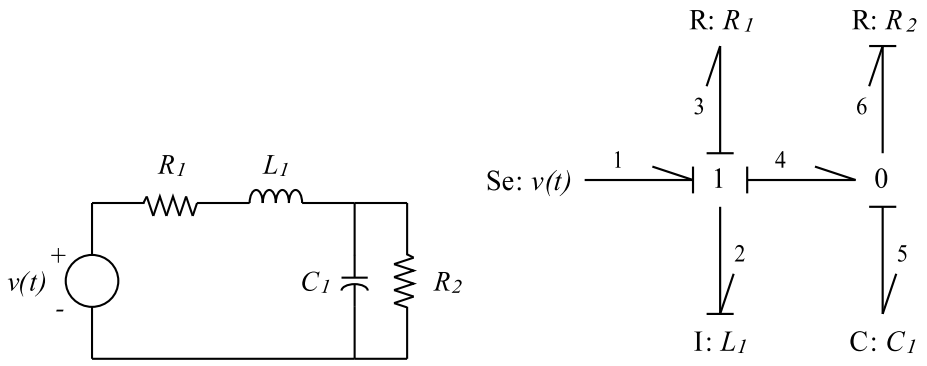
The chapter is organized as follows. We first describe the hypothesis generation procedure. Then, we describe briefly fault signatures, and formulate the concept of relative measurement orderings and illustrate how they are computed. We next discuss how fault isolation progresses as more measurement deviation events are provided, and how isolation may terminate. We conclude with a summary of the approach and comparisons to related approaches.

Hypothesis Generation

When the fault detector indicates the first measurement deviation, we obtain a `+` or `-` symbol representing whether the measurement is above or below its nominal value, respectively. The goal of hypothesis generation is, given this initial deviation σ_0, to produce a consistent set of initial fault candidates.

In order to do this, we utilize the backward propagation algorithm, `PropagateBackward`, of TRANSCEND [20]. The algorithm starts at a measurement vertex given the sign of the deviation, and propagates the deviation backwards through the TCG to include all fault parameters that are consistent with the observed deviation. The parameter change will be identified as a fault candidate if the direction of change of the incident vertex is consistent.

Example. Consider the circuit given in Fig. 28, which will be used as a running example throughout the chapter. The schematic and corresponding bond graph are shown Figs. 28a and 28b, respectively. Assume that a decrease in the measured current through R_2, or i_3^-, is observed. The annotated TCG for this circuit is given in Fig. 29. In the circuit, we consider $F = \{C_1^+, C_1^-, L_1^+, L_1^-, R_1^+, R_1^-, R_2^+, R_2^-\}$, for faults in the capacitor, inductor, and resistances. Propagating back in the TCG, the decrease in i_3 can be explained by f_6^- which in turn can be explained by R_2^+, given f_6 is positive. It can also be explained by e_6^-, which in turn can be explained by e_5^- and hence C_1^+. Propagating backwards further can link the change to parameter changes, R_1^+ and L_1^+. Thus the initial fault hypothesis set is $\{C_1^+, L_1^+, R_1^+, R_2^+\}$.



(a) Schematic. (b) Bond graph.

Figure 28: Example circuit.

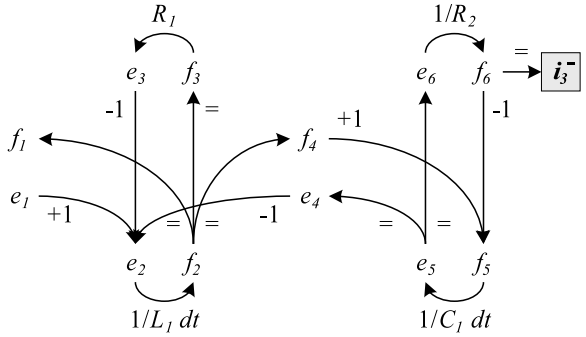


Figure 29: TCG for `PropagateBackward` example.

Prediction

The objective of the prediction phase is to determine the effects that faults will have on the measurements. These predicted effects are used to check if candidates are consistent with the observations. Predictions are represented formally as fault signatures, which qualitatively abstract measurement deviations caused by fault transients, and relative measurement orderings, which define temporal orders of the measurement deviations.

Fault Signatures

Abrupt faults generate transients in the dynamic system behavior. Assuming that the system output $y(t)$ is continuous and continuously differentiable except at the point of fault occurrence, t_f, the transient response at $t > t_f$ can be approximated by a Taylor series expansion [41, 45]:

$$y(t) = \sum_{n=0}^{\infty} y^{(n)}(t_f) \frac{(t - t_f)^n}{n!},$$

where $y^{(n)}(t_f)$ represents the nth derivative of the continuously differentiable signal $y(t)$ just after fault occurrence. If $|y^{(k+1)}|$ is bounded, the Taylor series expansion up to the k^{th} derivative is a good approximation of the true signal $y(t)$ for t close to t_f. For $t > t_f$, the residual, $r(t) = y(t) - \hat{y}(t)$, can then be approximated by

$$r(t) \quad = \quad \left(\sum_{n=0}^{\infty} y^{(n)}(t_f) \frac{(t-t_f)^n}{n!} \right) - \hat{y}(t).$$

This is the basis for establishing a signature for a fault transient, represented using the magnitude and derivative values of the residual signal. We abstract these magnitude and derivative values using the qualitative values `+`, `-`, and `0`, which imply an increase, decrease, or no change from the nominal behavior, respectively.

In general, a *fault signature* is defined as the qualitative value of zeroth- through kth-order derivative changes on a residual due to a fault occurrence. However, because only the magnitude and slope can be reliably measured, symbol generation only provides us with the immediate magnitude change and the observed slope. Therefore, we condense the full signatures to the magnitude change and the first nonzero derivative change, which eventually manifests as an first-order change that can be observed in the residual. For example, a seventh-order signature `000-+-+` becomes `0-`, and the signature `+-+-+-+` becomes `+-`. The set of possible restricted fault signatures is then given by {`+-`, `-+`, `++`, `--`, `+0`, `-0`, `0+`, `0-`, `00`}. The first symbol represents the immediate direction of abrupt change (a discontinuity) and the second symbol represents the slope that will be observed. The signatures `++` and `--` may occur for certain types of sensor faults, but for faults related to system parameters, they imply positive feedback, and hence, an unstable system, so are typically disregarded [45]. For `+0` and `-0`, the `0` slope symbol implies that the fault causes a jump but no subsequent change in the measurement. This occurs, for example, with sensor bias faults. The signature `00` represents no change due to the fault, i.e., the fault effect cannot reach the measurement because there is no causal relation. Given a measurement m, and deviation d, we write a signature as an event using m^d, e.g. m_1^{+-}. The definition of a fault signature is as follows.

Definition 13 (Fault Signature)**.** A *fault signature* for a fault f and measurement m is the qualitative magnitude and slope change in m caused by the occurrence of f, and is denoted by $\sigma_{f,m} \in \Sigma_{f,m}$. We denote the set of all fault signatures for fault f and measurements M as $\Sigma_{f,M}$, where $\Sigma_{f,M} = \bigcup_{m \in M} \Sigma_{f,m}$.

If the fault signature for a fault f and measurement m can be uniquely determined, then $\Sigma_{f,m}$ is a singleton. Ambiguities may arise in the qualitative arithmetic, however, resulting in a signature containing a $*$, which may manifest as either `+`, `-`, or `0`. So, in general, $\sigma_{f,m}$ may not be unique. For example, a `0*` may manifest as `0+` or `0-`.

Relative Measurement Orderings

The traditional TRANSCEND scheme uses only fault signatures to distinguish between faults. The order in which the measurements deviate is not taken into account when refining fault hypotheses. We define the notion of relative measurement orderings, which capture the intuition that fault effects will manifest in some parts of the system before others. For example, a fault occurring in one component will likely manifest first in that component and then in other components, if there are energy storage elements in the path between the sensors. If there are no energy storage elements, the relation is algebraic and no delay will be observed.

Definition 14 (Relative Measurement Ordering)**.** If the effects of fault f manifest in measurement m_i before measurement m_j then we can define a relative measurement ordering between m_i and m_j for fault f, denoted as $m_i \prec_f m_j$. We denote the set of all measurement orderings for a fault f and measurement set M as $\Omega_{f,M}$.

Relative measurement orderings can be derived from the system model given a particular mode. We describe how to generate orderings from the TCG, based on the notion of a *fault path*.

Definition 15 (Fault Path)**.** A fault path for a fault f and measurement m is a path in the TCG which begins at the fault f and ends at the measurement m. The set of all fault paths from f to m is denoted by $FP_{f,m}$.

Example. Consider a fault in R_2 for the TCG in Fig. 30, where f_2, e_5, and f_6 are measured. A fault path for R_2 to f_6 is $[f_6]$. Another is $[f_6 \xrightarrow{-1} f_5 \overset{1/C_1 dt}{\longrightarrow} e_5 \xrightarrow{=} e_6 \overset{1/R_2}{\longrightarrow} f_6]$.

Due to cycles associated with state loops, there are possibly an infinite number of fault paths, however, only a finite subset of these are important. The order of a fault path is defined as the number of temporal edges (i.e., edges with dt) in the path. A minimum order fault path is a path in $FP_{f,m}$ that contains the minimum number of temporal edges needed to reach m from f. More than one fault path of a specific order may exist for f and m, since there are often multiple paths from one vertex to another in the TCG.

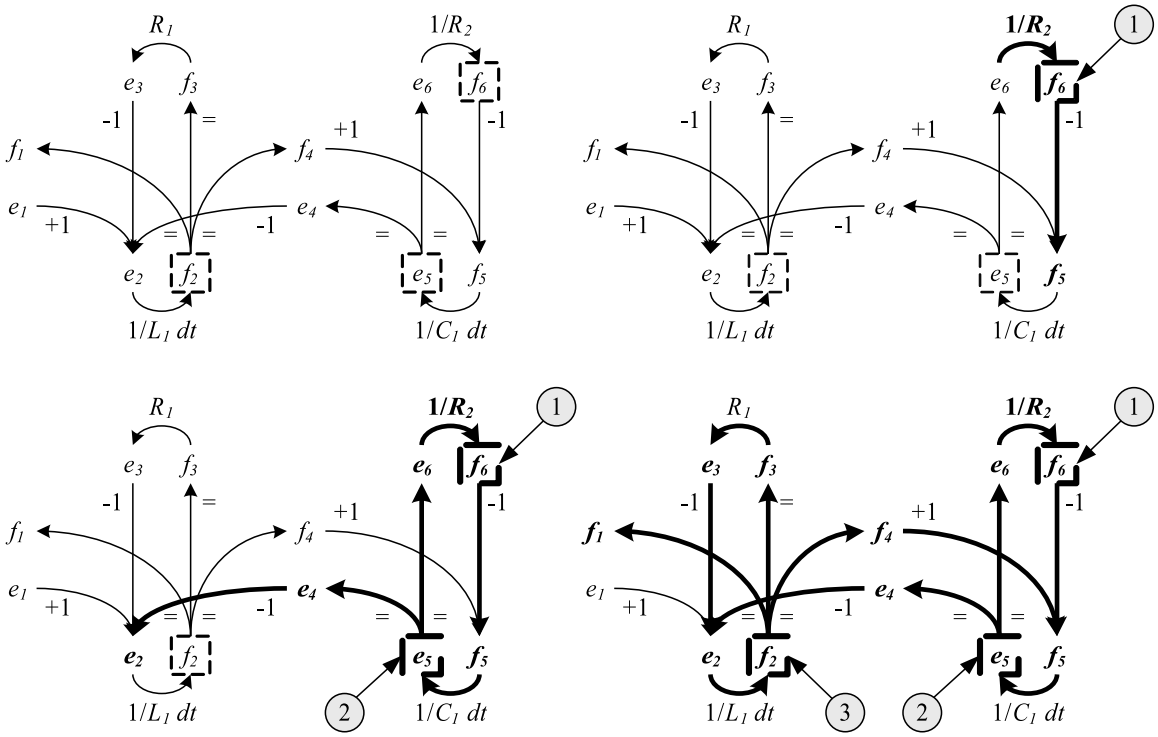


Figure 30: Derivation of relative measurement orderings for a fault in R_2.

Definition 16 (Minimum Order Fault Path Set)**.** The minimum order fault path set for f and m is the set of all minimum order fault paths, and is denoted as $FP^*_{f,m}$.

Example. Consider again a fault in R_2. The minimum order fault path set for f_6 is $\{[f_6]\}$. It has no temporal edges so its order is 0. For e_5, it is $\{[f_6 \xrightarrow{-1} f_5 \overset{1/C_1 dt}{\longrightarrow} e_5]\}$ with order 1, and for f_2, it is $\{[f_6 \xrightarrow{-1} f_5 \overset{1/C_1 dt}{\longrightarrow} e_5 \xrightarrow{=} e_4 \xrightarrow{-1} e_2 \overset{1/L_1 dt}{\longrightarrow} f_2]\}$ with order 2.

A fault path describes the temporal propagation of a fault to a specific measurement in the system. For a certain fault, there may be multiple fault paths leading to a measurement. Based on the Taylor series expansion of measured signals, we know the lowest order effect of a fault will manifest first [45]. So, only the minimum order fault path sets are useful in determining relative measurement orderings. For this purpose, we define a method for comparing fault paths.

Definition 17 (Temporal Subpath)**.** For $p \in FP_{f,m_i}$ and $p' \in FP_{f,m_j}$, p is a temporal subpath of p', denoted as $p \sqsubset p'$, if all temporal edges in p exist in p' in the same sequence, and the order of p is less than the order of p'.

Example. Given a fault in R_2, $[f_6] \sqsubset [f_6 \xrightarrow{-1} f_5 \overset{1/C_1 dt}{\longrightarrow} e_5] \sqsubset [f_6 \xrightarrow{-1} f_5 \overset{1/C_1 dt}{\longrightarrow} e_5 \xrightarrow{=} e_4 \xrightarrow{-1} e_2 \overset{1/L_1 dt}{\longrightarrow} f_2]$.

The following theorem shows how relative measurement orderings are derived from the TCG, based on the temporal subpaths, which can be related to transfer functions, and, therefore, be used to characterize delays.

Theorem 1. *If for every* $p' \in FP^*_{f,m_j}$ *there exists* $p \in FP^*_{f,m_i}$ *such that* $p \sqsubset p'$*, then we have* $m_i \prec_f m_j$*.*

Proof. In the signal flow graph for the TCG, let r_i be the measurement vertex corresponding to m_i, r_j the vertex for m_j, and r_f the successor vertex of the edge with fault parameter f. The transfer functions from r_f to r_i, $R_i(s)$ and from r_f to r_j, $R_j(s)$, can be derived. Assume for every $p' \in FP^*_{f,m_j}$ there exists $p \in FP^*_{f,m_i}$ such that $p \sqsubset p'$. Then each minimum order path from r_f to r_j must go through r_i or a vertex which can be expressed as $r_i \cdot G$, where G is some constant gain. $R_j(s)$ is a sum of terms which each correspond to different forward paths from r_f to r_j. Because lower order effects manifest first for a measurement, terms that correspond to forward paths of non-minimum order can be removed to produce $R'_j(s)$. Similarly, $R'_i(s)$ can be produced. Because every minimum order path from r_f to r_j goes through a vertex $r_i \cdot G$, $R'_i(s)$ must appear as a factor in each term of $R'_j(s)$, therefore $R'_j(s) = H(s)R'_i(s)$, where $H(s)$ is a proper transfer function. The order of m_i is less than the order of m_j by the definition of the $\sqsubset$ relationship, so the number of poles for $R'_i(s)$ must be less than the number for $R'_2(s)$. $H(s)$ must introduce more poles than zeros to $R'_j(s)$, and, therefore, $H(s)$ is strictly proper. From $H(s)$, we can discretize using the given sampling rate of the system to get $H(z)$. Since $H(s)$ is strictly proper, $H(z)$ is, therefore $r'_2(k) = f(r'_i(k-1))$. Since $r'_j(k)$ depends only on past values of $r'_i(k)$, with appropriately selected detection thresholds[1], a deviation resulting from fault f will appear first in m_i and then in m_j, thus $m_i \prec_f m_j$. □

Therefore, for a given fault f, we can say that it manifests in measurement m_i before measurement m_j if for all minimum order fault paths of m_j, there is a minimum order fault path for m_i that the fault will traverse before completing the traversal along the given fault path to m_j. Specifically, each of the fastest paths (i.e., the minimum order fault paths) to m_j must pass through m_i (or a variable algebraically related to it), after which an additional delay occurs, because the path order increases after passing through m_i. Hence, the transient due to the fault manifests later in time for m_j than for m_i. Thus, the deviation in m_i will be observed before the deviation m_j. Since we establish orderings based on common subpaths, we ensure that the time constants associated with

[1]This guarantees that for some time $|r_i(k)|$ will be greater than $|r_j(k)|$, after that time $|r_j(k)|$ may overtake $|r_i(k)|$ depending on the gain of $H(z)$. Therefore thresholds must be small enough such that deviations will cross them before that time.

the paths that produce the initial deviation on a measurement are shared. So, if the fault detectors are tuned with similar sensitivities, the orderings will hold.

Example. For example, consider the R_2^+ fault. Its minimum order fault path for f_6 is a temporal subpath of the minimum order fault path for e_5, which in turn is a temporal subpath for f_2. Thus, we can define $f_6 \prec_{R_2^+} e_5$, $f_6 \prec_{R_2^+} f_2$, and $e_5 \prec_{R_2^+} f_2$. However, if there was some other fault path of order 2 to f_2 that did not pass through e_5, we would not be able to establish the third ordering, because we would not know, in general, which path is faster, since the time constants associated with the paths are not shared. For L_1^+, $FP^*_{L_1^+,f_2} = \{[f_2]\}$, $FP^*_{L_1^+,e_5} = \{[f_2 \xrightarrow{=} f_4 \xrightarrow{+1} f_5 \xrightarrow{1/C_1 dt} e_5]\}$, and $FP^*_{L_1^+,f_6} = \{[f_2 \xrightarrow{=} f_4 \xrightarrow{+1} f_5 \xrightarrow{1/C_1 dt} e_5 \xrightarrow{=} e_6 \xrightarrow{1/R_2} f_6]\}$. We can define the orderings $f_2 \prec_{L_1^+} e_5$ and $f_2 \prec_{L_1^+} f_6$, but not one between e_5 and f_6, because they are of the same order so the temporal subpath relation will not hold.

Informally, two faults can be distinguished using orderings if there exist two measurements which deviate in some order for one fault, and in the opposite order for a second fault. Discrimination between faults using relative measurement orderings is based on the notion of temporal conflicts in the ordering relationships.

Definition 18 (Temporal Conflict)**.** A temporal conflict between ordering sets $\Omega_{f_i,M}$ and $\Omega_{f_j,M}$ exists if there are two measurements $m_k, m_l \in M$ such that $(m_k \prec_{f_i} m_l) \in \Omega_{f_i,M}$ and $(m_l \prec_{f_j} m_k) \in \Omega_{f_j,M}$.

The combined use of fault signatures and measurement orderings increases the discriminatory ability of the algorithms, because more information is utilized for refining candidates. We will show in Chapter VIII that for some systems, measurement orderings are actually necessary for the system to be diagnosable, i.e., signatures alone cannot diagnose the system.

Prediction Algorithm

The prediction algorithm computes fault signatures and relative measurement orderings given a parametric fault f, the set of measurements M, and a TCG, (V, E, L, l_E). `PropagateForward` extends the algorithm presented in [20] by computing measurement orderings. It is shown as Algorithm 1.

Propagation starts at the fault. If the fault is additive, and, hence, represented as a change in a source parameter, it is associated with a vertex, but if multiplicative, its parameter is associated with a set of edges (for TF and GY elements, the parameter will appear on multiple edges). The first step is to obtain the initial vertices from which to propagate and define the zeroth-order signature

Algorithm 1 $(\Sigma_{f,M}, \Omega_{f,M}) \leftarrow$ `PropagateForward`$(f, M, (V, E, L, l_E))$

```
V_pend ← ∅
set all Σ_f as unmarked
for all v ∈ V do
  TP_v ← ∅
if f is multiplicative then
  for all (v, l_f, v') ∈ E do
    σ_v'(0) ← consistent sign
    V_pend ← V_pend ∪ {(v', σ_v')}
else
  σ_v'(0) ← consistent sign
  V_pend ← V_pend ∪ {(v, σ_v')}
for all v ∈ V_pend do
  minOrder_v ← 0
while V_pend ≠ ∅ do
  v = PopFront(V_pend)
  for all (v, l, v') ∈ E do
    if σ_v' not determined to sufficient order then
      σ_v' ← UpdateSignature(v, σ_v, l, v', σ'_v)
      if l is instantaneous then
        PushFront(V_pend, v')
        if v' not visited then
          minOrder_v' ← minOrder_v
        if minOrder_v < minOrder_v' then
          TP_v' ← TP_v' ∪ TP_v
      else
        PushBack(V_pend, v')
        if v' not visited then
          minOrder_v' ← minOrder_v + 1
        if minOrder_v < minOrder_v' then
          TP_v' ← TP_v' ∪ {pl : p ∈ TP_v}
for all v ∈ V do
  set all unmarked derivatives in σ_v to 0
Ω_{f,M} ← ComputeOrderings(TP_v, M)
return (Σ_{f,M}, Ω_{f,M})
```

values for those vertices. Additive faults (e.g., sensor bias) are already represented by a vertex, so the change in vertex is positive if the fault is an increase, or negative if the fault is a decrease. For multiplicative faults, the initial signature for the vertex the edge points to is determined based on whether the predicted fault is an increase or decrease in parameter magnitude, and what the sign of nominal value of the variable is for that vertex. If it is positive, a parameter increase increases the variable and a parameter decrease decreases the variable. The opposite happens if the variable is negative.

The algorithm starts from the initial set of vertices and propagates their signatures to successor vertices in increasing derivative order. It also propagates sets of minimum order temporal paths encountered (consisting of only the temporal edges), which are used later to compute the measurement orderings. Propagating in increasing derivative order ensures these paths are built up correctly. Signatures are determined up to the sufficient order, which, under our two-symbol representation,

Function 2 $\Omega_{f,M} \leftarrow$ `ComputeOrderings`$(TPSet, M)$

for all $m_i, m_j \in M$ **do**
  if m_i reachable from f and m_j unreachable **then**
    $\Omega_{f,M} \leftarrow \Omega_{f,M} \cup \{m_i \prec_f m_j\}$
  else if $(\forall p_j \in TP_{m_j})(\exists p_i \in TP_{m_i}) p_i \sqsubset p_j$ **then**
    $\Omega_{f,M} \leftarrow \Omega_{f,M} \cup \{m_i \prec_f m_j\}$
return $\Omega_{f,M}$

means that the first two nonzero symbols in the signature must be computed for discontinuities, or the first nonzero symbol otherwise. The propagation stops when all reachable vertices have been determined to sufficient order. The signature propagation is handled by `UpdateSignature`, and is described in [20].

After propagation, the orderings are computed using the collected minimum order temporal paths. This procedure is given as Function 2. Measurement orderings are declared for reachable versus unreachable measurements, with respect to the fault. They are also declared when the condition of Theorem 1 is satisfied.

The signatures and orderings for the circuit of Fig. 13 are shown in Table 3. In the circuit, we define the measurements as $M = \{i_1, v_2, i_3\}$, or the current through L_1, the voltage across C_2, and the current through R_2, respectively. In the bond graph, these correspond to f_2, the flow of bond 2, e_5, the effort of bond 5, and f_6, the flow of bond 6. In deriving the signatures, we have assumed that variable values are nominally positive.

Example. Consider R_2^+. An increase in R_2 will cause an immediate decrease in f_6. Since all subsequent paths from f_6 to any other observed variable in the system contain some edge with a dt specifier (implying an integration), then deviations in these measurements will only be detected after f_6 deviates. The measured variable e_5 will deviate next with a first-order increase, i.e., `0+`. The change is opposite to the change in f_6 because of the -1 specifier in the path, which implies an inverse relationship. This change propagates back to f_6, yielding a complete signature of `+-`. The measured variable f_2 will deviate next because of the dt specifier on the path from e_5 to f_2, with a second-order decrease, i.e., `0-`. This will be eventually detected as a first-order change.

Event-based Fault Modeling

Fault signatures combined with relative measurement orderings provide event-based information for diagnosis. Therefore, we combine the notions of fault signatures and relative measurement orderings into an event-based framework, where significant measurement deviations are symbolically abstracted to events. For a specific fault, the combination of all fault signatures and relative mea-

Table 3: Fault Signatures and Relative Measurement Orderings for the Circuit

Candidate	i_1	v_2	i_3	Measurement Orderings
C_1^+	0+	-+	-+	$v_2 \prec i_1, i_3 \prec i_1$
C_1^-	0-	+-	+-	$v_2 \prec i_1, i_3 \prec i_1$
L_1^+	-+	0-	0-	$i_1 \prec v_2, i_1 \prec i_3$
L_1^-	+-	0+	0+	$i_1 \prec v_2, i_1 \prec i_3$
R_1^+	0-	0-	0-	$i_1 \prec v_2, i_1 \prec i_3$
R_1^-	0+	0+	0+	$i_1 \prec v_2, i_1 \prec i_3$
R_2^+	0-	0+	-+	$v_2 \prec i_1, i_3 \prec i_1, i_3 \prec v_2$
R_2^-	0+	0-	+-	$v_2 \prec i_1, i_3 \prec i_1, i_3 \prec v_2$

surement orderings yields all the possible ways a fault can manifest. Our event set is then the set of possible measurement deviations. We denote each of these possibilities as a *fault trace*.

Definition 19 (Fault Trace)**.** A *fault trace* for a fault f over measurements M, denoted by $\lambda_{f,M}$, is a string of length $\leq |M|$ that includes, for every $m \in M$ that will deviate due to f, a fault signature $\sigma_{f,m}$, such that the sequence of fault signatures satisfies $\Omega_{f,M}$.

Example. Consider C_1^+ for measurements $M = \{i_1, v_2, i_3\}$. To simplify the notation we will drop the M subscript in our examples. $\lambda_{C_1^+} = v_2^{-+} i_3^{-+} i_1^{0+}$ is a valid fault trace, but $\lambda_{C_1^+} = i_1^{0+} v_2^{-+} i_3^{-+}$ is not because the measurement deviation sequence does not satisfy $\Omega_{C_1^+}$.

Note that the definition implies that fault traces are of maximal length, i.e., a fault trace includes deviations for all measurements affected by the fault. We group the set of all fault traces into a *fault language*. The *fault model*, defined by a *finite automaton*, concisely represents the fault language.

Definition 20 (Fault Language)**.** The *fault language* of a fault $f \in F$ with measurement set M, denoted by $L_{f,M}$, is the set of all fault traces for f over measurements M.

Definition 21 (Fault Model)**.** The *fault model* for a fault $f \in F$ with measurement set M, is the finite automaton that accepts exactly the language $L_{f,M}$, and is given by $\mathcal{L}_{f,M} = (S, s_0, \Sigma, \delta, A)$ where S is a set of states, $s_0 \in S$ is an initial state, Σ is a set of events, $\delta : S \times \Sigma \to S$ is a transition function, and $A \subseteq S$ is a set of accepting states.

The finite automata representation allows for the composition of the fault signatures and relative measurement orderings into fault models. The possible fault signatures $\Sigma_{f,m}$ can be represented as a finite automaton with event set $\Sigma_{f,m}$, shown in Fig. 31 (left), for the case where $\Sigma_{f,m}$ is a singleton. It consists of only the single event corresponding to the fault signature. In general, multiple edges for each $\sigma_{f,m} \in \Sigma_{f,m}$ are needed going from the first state of the automaton to the final state. This represents the constraint that a measurement's deviation is only observed once. Also, each

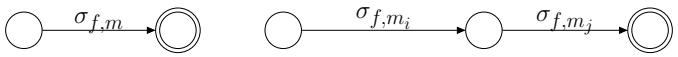
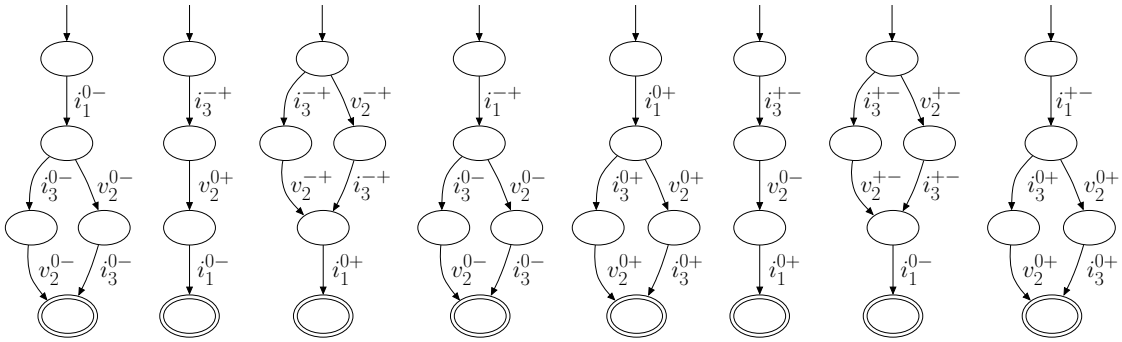


Figure 31: Fault signature finite automaton representation (left) and relative measurement ordering finite automaton representation (right).



(a) $\mathcal{L}_{R_1^+}$ (b) $\mathcal{L}_{R_2^+}$ (c) $\mathcal{L}_{C_1^+}$ (d) $\mathcal{L}_{L_1^+}$ (e) $\mathcal{L}_{R_1^-}$ (f) $\mathcal{L}_{R_2^-}$ (g) $\mathcal{L}_{C_1^-}$ (h) $\mathcal{L}_{L_1^-}$

Figure 32: Fault models for the faults of the circuit.

relative measurement ordering, $m_i \prec_f m_j$, with associated signature sets Σ_{f,m_i} and Σ_{f,m_j}, can be represented as an automaton with event set $\Sigma_{f,m_i} \cup \Sigma_{f,m_j}$, shown in Fig. 31 (right), for the case where Σ_{f,m_i} and Σ_{f,m_j} are singletons. The automaton consists of the associated signatures in the determined ordering. The following lemma formalizes the composition of these automata.

Lemma 1. *For fault model for fault f, $\mathcal{L}_{f,M}$, is the synchronous product of the individual finite automata for all $\sigma_{f,m} \in \Sigma_{f,M}$ and all $(m_i \prec_f m_j) \in \Omega_{f,M}$.*

Proof. Since the synchronous product must accept fault traces that obey all individual ordering constraints and includes all measurement deviation events for the fault, it accepts all valid measurement deviation sequences, i.e., all $\lambda \in L_{f,M}$, and no others. □

Example. Fig. 32 shows the fault models for the circuit example. Accepting states are denoted by a double circle. Consider R_1^+. Its orderings specify that i_1 must deviate before v_2 and i_3. Therefore, i_1^{0-} is first, followed by v_2^{0-} and i_3^{0-} in either order.

Fault models are an alternative representation of fault signatures and measurement orderings that combine the two forms of diagnostic information into a formal event-based framework. Clearly, forming fault models requires additional computation after signatures and orderings are computed. Therefore, it is not immediately clear what the computational advantage of fault models is. We use the formal framework developed in this section in the rest of the chapter to precisely describe our methodology for consistency-based diagnosis. The algorithms do not explicitly use fault models,

but we describe in Chapter VII how fault models form the basis to a compilation-based technique for efficient online diagnosis, and how the formal framework they provide can be used to formulate notions of diagnosability and obtain systematic methods for deciding if a system is diagnosable.

The construction of fault models is based on the assumption that signatures and orderings can be precomputed. As we observed in the previous section, however, the effect of faults depends on the signs of the variables in the system. For example, a resistance increase, R^+, causes the current through the resistor to decrease if the current is positive, but increase if negative, because it becomes less negative. If the variable signs can change over time, they can be accounted for by selecting the appropriate fault models. Since the prediction algorithm takes into account variable sign, then the correct signatures will be computed at fault detection. If the sign is not taken into account, then we would isolate R^- instead of R^+, but by checking the sign of the immediate variable affected by R, we can determine which of these is the correct candidate.

Candidate Tracking

Hypothesis generation produces the set of fault candidates consistent with the initial measurement deviation, and prediction computes future effects these candidates will have on the measurements. Progressive monitoring uses the predictions (i.e., signatures and orderings, or, equivalently, fault models) to match candidates to new measurement deviations, but must refine candidates when a new measurement deviation occurs. Measurement deviations may implicate new faults. We extend the progressive monitoring algorithms of TRANSCEND to use both fault signatures and relative measurement orderings, and to isolate multiple parametric faults.

Progressive monitoring is shown as Algorithm 3. It begins with the initial diagnosis d_0, consisting of single-fault candidates obtained using backward propagation for event σ_0. Since backward propagation uses only the initial sign of the deviation and does not use measurement orderings in selecting initial candidates, we first predict the effects of the faults. We then check consistency using the full signatures and measurement orderings. If consistent with the predictions, the candidate is placed in d.

After initializing d, `CandidateTracking` waits for events to occur. When it receives a measurement deviation event, it must refine the current candidates. This consists of checking whether the current candidates are consistent with the new information, and if not, generating new, more complex candidates that are consistent. In general, hypothesis refinement works as follows. Given a current diagnosis d_i, and the measurement deviation event, σ_{i+1}, we generate a new diagnosis, d_{i+1}.

Algorithm 3 $d \leftarrow$ `CandidateTracking`(d_0, σ_0)

```
d ← ∅
for all {f_i} ∈ d do
   P_{f_i} ← PropagateForward(f_i, M, TCG)
   if f_i consistent with σ_0 then
      d ← d ∪ {{f_i}}
while termination condition is false do
   wait for σ
   d ← RefineHypotheses(d, σ)
return d
```

For each candidate of the old diagnosis (d_i), we check consistency with the new deviation (σ_{i+1}). If consistent, we retain the candidate. If inconsistent, we reject the candidate. However, there may be a more complex candidate, constructed from the rejected candidate, that is consistent, and we need to explore the resultant possibilities. We add newly created candidates which are consistent to the new diagnosis (d_{i+1}). We then prune the new diagnosis to make it minimal and to eliminate candidates above the candidate cardinality limit l (Assumption 1).

Because multiple fault diagnosis builds on single fault diagnosis, we first explain hypothesis refinement for single faults, and then for multiple faults. The `RefineHypotheses` algorithm has a different implementation in each case, although setting $l = 1$ in the multiple fault algorithm reduces it to the single fault algorithm.

Refining Single Fault Hypotheses

First, let us take the simplest case of single fault diagnosis (SFD) in a continuous system. Since candidates can only be single faults, we simplify our representation of a candidate c to a single fault rather than a set of faults, and omit the mode, i.e., $c = f$.

When a measurement deviation occurs, we check if each candidate is consistent. Consistent candidates are retained and inconsistent candidates dropped. We need to be able to define how a candidate is consistent, i.e., what are the possible traces that the candidate may be linked to. Consistency is defined formally through the use of a *language* for a candidate. For SFD in continuous systems, these languages correspond directly to the fault languages. Given a complete fault trace, we can check which fault languages contain the trace, and this provides our set of consistent faults, i.e., our diagnosis. Clearly, traces are built up over time as measurements deviate. Therefore, we check consistency in an incremental manner. Given a partial trace, the set of consistent faults is given by those whose language contains a trace that is an extension of the given partial trace. In simpler terms, given a new measurement deviation, a candidate (i.e., fault) is consistent if the signature

Algorithm 4 $d_{i+1} \leftarrow$ `RefineHypotheses`(d_i, σ_{i+1}) (Single Fault Diagnosis)

$d_{i+1} \leftarrow \varnothing$
for all $f_i \in d_i$ **do**
  if f_i consistent with $\lambda_i \sigma_{i+1}$ **then**
    $d_{i+1} \leftarrow d_{i+1} \cup \{f_i\}$
return d_{i+1}

of the fault on that measurement matches, and the sequence of measurement deviations up to the current point in time satisfies the measurement orderings.

Matching the orderings is straightforward, but matching the signatures has two special cases. The first is the `*` symbol that may be produced during forward propagation due to an ambiguity in the qualitative arithmetic. The `*` symbol may manifest as `+`, `-`, or `0` in the symbol generation. So the `*` symbol will match any observed deviation. For example, an observed `0+` will match a predicted `0*`. As described previously, the event-based fault models represent both possible deviations. The second special case is a `0` observation for the slope. In our symbol generation procedure, a `0` computation for the slope occurs when the slope cannot be determined to be `+` or `-` within the slope window. This may happen either because the slope is truly zero or very small over the window. But, we assume that if a `0` slope is calculated, then the observed discrepancy is a discontinuity. So, for example, an observed `-0` would not match with a predicted effect of `0-` but would match with `-0` and `-+`.

`RefineHypotheses` for SFD is shown as Algorithm 4. The new diagnosis, d_{i+1}, is formed from the old diagnosis d_i, by retaining consistent candidates and dropping inconsistent candidates. So, consistency is checked in an incremental manner. We denote by λ_i the trace from event σ_0 up to σ_i. Given the new deviation, σ_{i+1}, the new trace is $\lambda_i \sigma_{i+1}$. Before the new deviation, all faults in the diagnosis are known to be consistent with λ_i. When a new deviation occurs, we need to make sure each fault is still consistent. We can do this by checking that the predicted signature matches the observed effect and that the orderings are satisfied. Since we know orderings have been satisfied up to the current point, then to check if the new deviation is consistent, we only need to check it against measurements that have not yet deviated. That is, we can ignore measurements which have already deviated in checking the orderings. So, in a sense, we are projecting out already deviated measurements, and checking consistency with the rest of the measurements for the signatures and orderings. After all measurements deviate, the diagnosis will contain only faults for which the final trace belongs to their fault language or can be extended to a larger trace in their language.

Example. Assume we observe i_1^{0-} as the first deviation for the circuit of Fig. 28. Both R_1^+ and C_1^- match the signature. However, the prediction for C_1^- is that either v_2 or i_3 will deviate before i_1, so C_1^- is not consistent. The prediction for R_1^+ is that i_1 will deviate first, so R_1^+ is consistent.

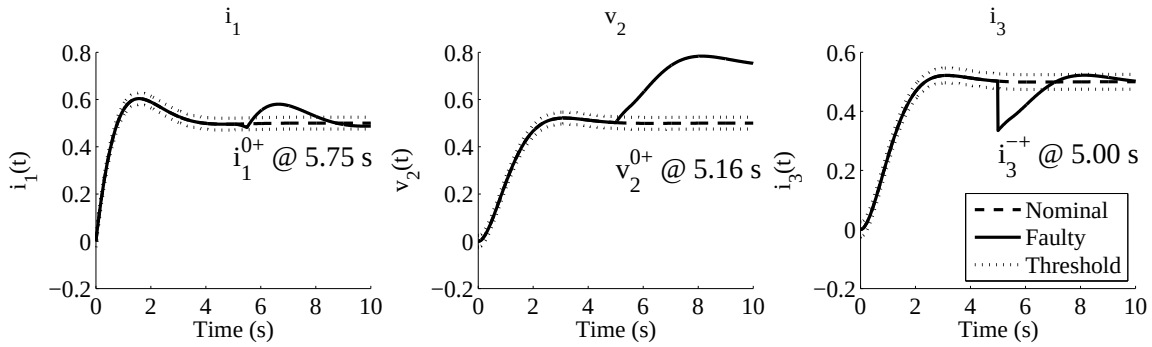


Figure 33: Faulty versus nominal behavior for the circuit measurements. R_2^+ (50% increase) is injected at 5.00 s and R_1^- (50% decrease) at 5.50 s.

We should next observe v_2^{0-} and i_3^{0-} in either order, thus the full fault trace will be $i_1^{0-}v_2^{0-}i_3^{0-}$ or $i_1^{0-}i_3^{0-}v_2^{0-}$.

Refining Multiple Fault Hypotheses

We now take the case of multiple fault diagnosis (MFD) in a continuous system. For this, we omit the mode to simplify our representation of a candidate c, i.e., $c = F_i$, where $F_i \subseteq F$.

Checking consistency for MFD is more complex than SFD due to (i) fault masking, which is a function of fault magnitudes, which are unknown, and (ii) relative time of individual fault occurrences, which are also unknown. Therefore, the *language* of a candidate for MFD is actually formed by some sort of composition of the possible traces of the faults it contains. We need to analyze the effect of multiple faults on the signatures and orderings to correctly characterize the candidate language and, thus, consistency, for MFD.

Example. Assume R_1^- and R_2^+ both occur, and R_2^+ happens before R_1^-, as shown in Fig. 33. In this example, all nominal parameter values are chosen to be 1. Here, the fault trace is $i_3^{-+}v_2^{0+}i_1^{0+}$. This sequence of measurement deviations is not consistent with any single fault. It matches with R_2^+ up until the third deviation. If we made the single fault assumption, R_2^+ would be dropped. For MFD, we must look for additional explanations beyond single faults that can cause the observed behavior.

What we find is that when multiple faults occur, a conservative estimate of the deviation on a single measurement can be expressed as the union of the individual signatures. So, with R_1^- and R_2^+ occurring together, we may see either `0+` or `0-` on i_1, only `0+` on v_2, and `0+` or `-+` on i_3. However, these must be constrained by the measurement orderings. For example, a fault trace such as $i_3^{0+}v_2^{0+}i_1^{0-}$ is not possible if these two faults occur together, because even though the signatures

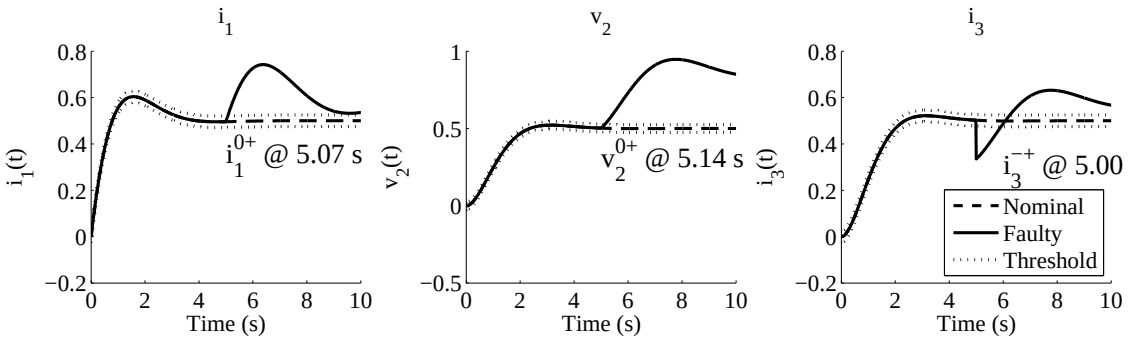


Figure 34: Faulty versus nominal behavior for the circuit measurements. R_2^+ (50% increase) is injected at 5.00 s and R_1^- (75% decrease) at 5.00 s.

all belong to either R_1^- or R_2^+, the sequence of the signatures does not match, i.e., orderings are necessary for correctly characterizing the possible traces resulting from multiple faults. Either i_1^{0+} (from R_1^-) or i_3^{-+} (from R_2^+) should have occurred as the first deviation.

This concept relates back to the derivation of measurement orderings from the TCG. Assume a fault f_1 produces the fault trace $m_1^{0+}m_2^{0+}m_3^{0+}$ and a fault f_2 produces the fault trace $m_3^{0-}m_2^{0-}m_1^{0-}$. We cannot see a trace such as $m_1^{0+}m_2^{0-}m_3^{0+}$. The first deviation (from f_1) is possible, but the second (from f_2) is not. This is because the orderings imply that the fastest path from f_2 to m_2 is through m_3, so in order to see f_2's effect on m_2, m_3 would have to deviate first. If f_2 had occurred, we would have to see its deviation on m_3 or some other deviation on m_3 from another fault that masks f_2's effect before we can see f_2's deviation on m_2.

So, because of masking, the signatures are found in the union of the signatures of the individual faults, but must be constrained by orderings. Relative time of fault occurrence matters as well, and in conjunction with fault masking, implies that multiple faults occurring together can manifest in many different ways.

Example. Fig. 34 shows again R_1^- and R_2^+ both occurring, but this time simultaneously, and with R_1^- having a larger magnitude. Now, the observations are the same but the sequence changes. With these magnitudes and fault occurrence times, the trace changes to $i_3^{-+}i_1^{0+}v_2^{0+}$. Similarly, we can imagine R_1 occurring later and not even observing its effects as initial deviations on the measurements. Or, R_2^+ may occur at a larger magnitude, and so we observe i_1^{0-} instead of i_1^{0+}.

Clearly, relative times of fault occurrence and fault magnitudes play a large role. Our approach can only be viewed as a *best effort* paradigm, because we can only hypothesize the occurrence of faults for which we actually see consistent deviations. Just as with single fault diagnosis, this is a conservative approach. We will produce candidates for which we see consistent deviations,

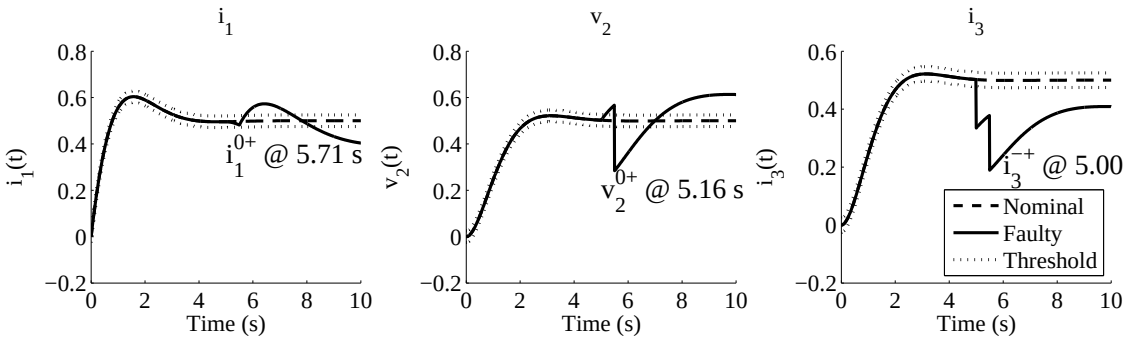


Figure 35: Faulty versus nominal behavior for the circuit measurements. R_2^+ (50% increase) is injected at 5.00 s and C_1^+ (100% increase) at 5.50 s.

however, we will not explicitly generate nonminimal candidates for which minimal explanations are consistent. For example, if two faults occur together, we can never determine with certainty that both faults occurred if the effects are consistent with just one of the faults occurring by itself. Our minimal diagnoses represent the possible certain candidates that we can obtain, but, as described in Chapter III, *cover* nonminimal candidates which cannot be determined with certainty. Therefore, we still end up with a diagnosis that represents all consistent possibilities.

Since multiple faults can produce many different possible measurement deviation sequences, the distinguishability of the diagnosis algorithms reduces. That is, different combinations of faults can produce the same traces, even if the single faults by themselves all produce unique traces. This is demonstrated by the following example.

Example. Consider R_2^+ and C_1^+ occurring together, as shown in Fig. 35. The same fault trace as the first example (Fig. 33) is observed, i.e., $i_3^{-+} v_2^{0+} i_1^{0+}$. Therefore this particular fault trace matches both candidate $\{R_2^+, C_1^+\}$ and candidate $\{R_2^+, R_1^-\}$, even though the individual faults are each distinguishable from each other.

These examples give us an intuitive idea of how to define the language of multiple-fault candidate, i.e., what possible traces multiple faults can produce. As discussed, the measurements that deviated up to a particular point play a large role in determining the possible traces. In the previous example, i_3^{-+} forms the start of a fault trace for both C_1^+ and R_2^+. The next deviation, v_2^{0+}, continues the trace for R_2^+ but not for C_1^+. The third deviation, i_1^{0+}, continues a trace for C_1^+, although the deviation in v_2 that preceded it did not match C_1^+. So at each point, the consistent "new" faults are those that match the deviation with respect to the previously deviated measurements projected out. We can formalize the notion of *measurement projection* on a trace as follows.

Definition 22 (Measurement Projection)**.** A *measurement projection* for a trace over measurements M_i, P_{M_i}, is defined as

$$\begin{aligned} P_{M_i}(\epsilon) &= \epsilon \\ P_{M_i}(\sigma_m) &= \begin{cases} \epsilon & m \notin M_i \\ \sigma_m, & m \in M_i \end{cases} \\ P_{M_i}(\lambda\sigma_m) &= P_{M_i}(\lambda)P_{M_i}(\sigma), \end{aligned}$$

where ϵ is the empty trace. The *measurement projection* for a fault language L is defined as $P_{M_i}(L) = \{\lambda : P_{M_i}(\lambda') \text{ where } \lambda' \in L\}$.

Essentially, $P_{M_i}(\lambda)$ returns a version of λ without the events for measurements in $M - M_i$. For example, $P_{\{i_1,i_3\}}(i_3^{-+}v_2^{0+}i_1^{0-}) = i_3^{-+}i_1^{0-}$. We are interested in projecting out measurements that have already deviated, so if M_i is the set of measurements that have already deviated, we use P_{M-M_i}. Note also that $L_{f,M-M_i} = P_{M-M_i}(L_{f,M})$, i.e., the fault language computed with a subset of the measurements $M - M_i$ (those that have not yet deviated) is the same as that fault language with the measurements in M_i (those that have deviated) projected out.

Using this concept, we can now begin to define the candidate language for MFD in continuous systems. We start by defining a *candidate trace*, which extends our earlier notion of a fault trace and is based on the notion of a *prefix*.

Definition 23 (Prefix)**.** A trace λ_i is a *prefix* of trace λ_j, denoted by $\lambda_i \sqsubseteq \lambda_j$, if there is some (possibly empty) sequence of events λ_k that can extend λ_i such that $\lambda_i\lambda_k = \lambda_j$.

Definition 24 (Candidate Trace)**.** An event trace $\lambda = \sigma$ is a *candidate trace* for $c = F_i$, if $\sigma \sqsubseteq \lambda' \in L_{f,M}$ for some $f \in F_i$. An event trace $\lambda = \lambda_i\sigma_{i+1}$ is a *candidate trace* for $c = F_i$, if λ_i is a candidate trace for F_i, and $\sigma_{i+1} \sqsubseteq \lambda' \in L_{f,M-M_i}$ for some $f \in F_i$. A candidate trace for c is denoted as λ_c.

We are only concerned with *maximal* traces, i.e., those for which all measurements that will deviate for the faults of the candidate have deviated (as with fault traces).

Definition 25 (Maximal Candidate Trace)**.** A candidate trace λ_c for $c = F_i$ is *maximal* if for all $f \in F_i$, $L_{f,M-M_i} = \varnothing$, where M_i is the set of deviated measurements for λ_c.

Now, we can define the language of a candidate $c = F_i$, $L_{c,M}$, as the set of maximal candidate traces for c.

Definition 26 (Candidate Language)**.** The *candidate language* for candidate $c = F_i$ and measurements M, denoted as $L_{c,M}$, is the set of all maximal candidate traces λ_c.

Example. Consider R_2^+ and C_1^+ occurring together, as shown in Fig. 35 with trace $i_3^{-+} v_2^{0+} i_1^{0+}$. If we inspect the fault models for these two faults, we can see that this corresponds to the first two events in $\mathcal{L}_{R_2^+,q_1}$ and the first and third events in $\mathcal{L}_{C_1^+,q_1}$, where the second event is skipped over. After the first event, we advance in both fault models, i.e., the deviation may be from either fault. Then, we project out i_3, and the next event should be either v_2^{0+} (from R_2^+) or v_2^{+-} (from C_1^+). The next event is v_2^{0+} which must have been caused by R_2^+. We then project out both i_3 and v_2, and the next event should be i_1^{0-} (from R_2^+) or i_1^{0+} (from C_1^+). The next event is i_1^{0+}, which must have been caused by C_1^+.

The candidate language for a set of faults essentially contains all traces for which at each point in the trace, there is a fault in the set whose abbreviated fault language (that with deviated measurements projected out) contains a trace that starts with the new event. Clearly, this maps easily back to incremental consistency checking, as with SFD. Before the new deviation, all fault sets in the diagnosis are known to be consistent. When a new deviation occurs, we need to make sure each candidate is still consistent. As the previous examples show, when new faults manifest, they must respect the signatures and measurement orderings for the fault with respect to measurements that have not already deviated due to other faults. So we can determine the faults that are consistent with a new deviation, σ_{i+1}, by checking if the event forms the start of a valid trace in the fault languages with deviated measurements, M_i (i.e., measurements for events in the current trace, λ_i), projected out. We call this set of faults the *hypothesis set*.

Definition 27 (Hypothesis Set)**.** For a fault set $F = \{f_1, f_2, \ldots, f_n\}$ and deviated measurements M_i, with fault languages $\{L_{f_1,M-M_i}, L_{f_2,M-M_i}, \ldots, L_{f_n,M-M_i}\}$, the *hypothesis set* $h_{F,M_i}(\sigma_{i+1})$ for the new event σ_{i+1} is computed as the set of all $f \in F$ such that there exists some $\lambda \in L_{f,M-M_i}$ where $\sigma_{i+1} \sqsubseteq \lambda$.

A hypothesis set contains a fault if the language for the fault over measurements which have not yet deviated ($M - M_i$) is consistent with σ_{i+1}. A fault will be consistent if σ_{i+1} forms the start of a valid fault trace for its abbreviated language ($L_{f,M-M_i}$). The hypothesis set essentially represents the set of "new" faults that may have occurred, although it may contain faults that are already in the current diagnosis.

Hypothesis sets are used in constructing new diagnoses incrementally. A previous diagnosis represents the set of candidates consistent up to the current point in time. Given a new event, candidates that are still consistent can be retained. However, candidates which are no longer consistent can now be explained by a new version of the candidate containing at least one more fault in the

hypothesis set. Each candidate represents a logical and of faults, and both d and h represent the logical or of these candidates. So, we can construct the new diagnosis (i.e., what is true now) as the old diagnosis d, *anded* with the hypothesis set h.

Definition 28 ($\wedge$ Operator)**.** For diagnosis d and hypothesis set h, d in conjunction with h, denoted as $d \wedge h$, is defined as $\{F_i \cup \{f_j\} : F_i \in d, f_j \in h\}$.

As a second step, we need to prune the diagnosis by removing (i) any candidates with cardinality above the limit l (Assumption 1), (ii) any candidate which is a superset of another (to make the diagnosis minimal), and (iii) any candidate which violates fault persistency (Assumption 4). A candidate that contains, e.g., both R_1^+ and R_1^- says that the same parameter changed twice, so violates persistency. We define the overall operation using $\wedge_l$.

Definition 29 ($\wedge_l$ Operator)**.** For diagnosis d and hypothesis set h, $d \wedge_l h$ is defined as $\{F_i \in d \wedge h : |F_i| \leq l, (\neg \exists F_j \in d \wedge h, F_j \subset F_i), F_i$ satisfies Assumption 4$\}$.

If we choose the candidate cardinality limit $l = 1$, then the process reduces to SFD, and we can obtain the new diagnosis simply by intersecting the old diagnosis with the hypothesis set. For SFD, though, the hypothesis set does not even need to be derived in the first place, because we are only concerned with faults in the hypothesis set that are already in the current diagnosis, i.e, we only eliminate candidates from the current diagnosis, and never add to it.

Example. Assume we observe i_3^{-+} first, giving an initial diagnosis of $\{R_2^+, C_1^+\}$. Since this is our first deviation, our hypothesis set is also $\{R_2^+, C_1^+\}$. Say that next we see v_2^{0+}. Now, ignoring i_3, we obtain the hypothesis set $\{R_2^+\}$. Given our previous diagnosis, our new diagnosis is our old diagnosis anded with our hypothesis set. In this case, we obtain $\{R_2^+, C_1^+ R_2^+\}$, or, as a minimal diagnosis, $\{R_2^+\}$. If we next observe i_1^{0+}, then ignoring i_3 and v_2, we get a hypothesis set of $\{R_1^-, C_1^+\}$. Anding this with our previous diagnosis, we obtain $\{R_2^+ R_1^-, R_2^+ C_1^+\}$, and this agrees with Figs. 33 and 35.

The incremental consistency checking has similarities with GDE [3]. For example, consider the fault signatures in Table 4 and the candidate lattice given in Fig. 36. The lattice represents the space of possible candidates. In GDE, observations yield conflicts, which are used to generate new diagnoses. A conflict is defined as a set of assumptions which cannot all be true, and thus support a symptom (e.g., $\overline{a_1 \wedge a_2 \wedge a_3}$). In our approach, we obtain direct symptom to fault knowledge for each single fault, i.e., fault signatures and measurement orderings, or fault models. So, instead of representing the *negated and* of assumptions, we represent the *or* of *negated* assumptions. This just applies De Morgan's rule, so it is equivalent. Our negated assumptions are faults, i.e., an

Table 4: Lattice Example Fault Signature Table

Fault	m_1	m_2	m_3
f_1	0+	0+	0+
f_2	0+	0+	0−
f_3	0+	0−	0+
f_4	0−	0−	0−

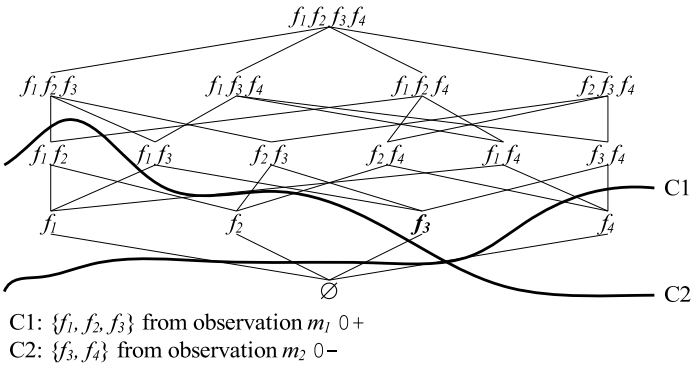


Figure 36: Minimality in the Lattice Representation

assumption is that a fault has not occurred, so we obtain $\overline{a_1 \wedge a_2 \wedge a_3} \equiv \overline{a_1} \vee \overline{a_2} \vee \overline{a_3} \equiv f_1 \vee f_2 \vee f_3$. Thus conflicts are equivalent to the hypothesis sets of our approach. The difference is that GDE performs this operation in the context of static models, but we generate ours correctly in a dynamic setting.

In the lattice of Fig. 36, we may get a conflict C1 (or hypothesis set) of $\{f_1, f_2, f_3\}$. So, each fault by itself is consistent. The C1 line of the lattice then rules out f_4 by itself as a consistent candidate. The new maximal diagnosis is given by all the candidates above the line. Given a conflict C2 of $\{f_3, f_4\}$ allows us to eliminate candidates which do not contain either f_3 or f_4. Given both conflicts, the maximal diagnosis is defined by the intersection of the two maximal diagnoses. In this case, the minimal diagnosis is simply $\{f_3\}$. The updating of the current diagnosis using conflicts is equivalent to our updating of the current diagnosis using hypothesis sets. The main difference between the approaches is our direct application to dynamic systems, through the use of event-based information in forming the hypothesis sets.

A key contribution of the GDE approach is the notion of a *minimal cover*. Using conflicts (or hypothesis sets) systematically obtains minimal diagnoses, because with each new conflict, we are only increasing our current candidates by at most one additional fault. The `RefineHypotheses` algorithm for MFD, shown as Algorithm 5, takes advantage of minimality. If all the candidates in the current diagnosis can explain the new deviation, we retain the previous diagnosis, because it

Algorithm 5 $d_{i+1} \leftarrow$ `RefineHypotheses`(d_i, σ_{i+1}) (Multiple Fault Diagnosis)

```
d_{i+1} ← ∅
if all c ∈ d_i consistent with λ_i σ_{i+1} then
    d_{i+1} ← d_i
else
    h_{i+1} ← ∅
    F_{i+1} ← PropagateBackward(σ_{i+1}, TCG)
    for all f_{i+1} ∈ F_{i+1} do
        P_{f_{i+1},M} ← PropagateForward(f_{i+1}, M, TCG)
        if f_{i+1} consistent with σ_{i+1} for M − M_i then
            h_{i+1} ← h_{i+1} ∪ {f_{i+1}}
    d_{i+1} ← d_i ∧_l h_{i+1}
return d_{i+1}
```

is minimal. If not, we obtain the hypothesis set, anding it with the previous diagnosis to obtain the new diagnosis. We do this by first using `PropagateBackward` to obtain an initial set of single faults for the new deviation, but since `PropagateBackward` does not use the full signature or use orderings, it may generate faults that are inconsistent as single faults, given the full information. We compute full predictions using `PropagateForward`, and use these to check consistency for the set of measurements that have not yet deviated ($M - M_i$), where M_i denotes the set of measurements for events in λ_i. We then compose the hypothesis set h_{i+1} with the old diagnosis d_i using $\wedge_l$, which takes the logical and, and prunes the resultant diagnosis.

We achieve minimality through two operations. First, we only augment a candidate if it cannot explain the current deviation. So in the previous example, we had a diagnosis of $\{R_2^+, C_1^+\}$ and obtained v_2^{0+}. This is still consistent with the candidate R_2^+, so we do not seek out other candidates which are consistent but not minimal, such as $R_2^+ R_1^+$. Since we see no direct evidence that R_1^+ has occurred, we do not consider it. After composing the diagnosis with the hypothesis set, we obtained $\{R_2^+, C_1^+ R_2^+\}$. Our second operation is to prune this to make it minimal. We have seen no direct evidence that C_1^+ and R_2^+ must have occurred together (since there is an alternate, minimal explanation, i.e., that R_2^+ by itself occurred), so remove it from the diagnosis. As described in Chapter III, we can do this because the candidate $\{R_2^+\}$ covers the candidate $\{C_1^+, R_2^+\}$.

Although we employ a fault size limit of l, we are actually limited in what we can distinguish by the number of measurements, $|M|$. Each new measurement deviation can only add one fault to a given candidate, so a candidate cannot grow in size beyond $|M|$. Therefore our minimal diagnosis will never have candidates above size $|M|$.

Terminating Fault Isolation

Candidate tracking stops when the termination condition is true. Determining such a condition is an important issue, and nontrivial for MFD. In SFD, we can terminate whenever we have a unique candidate left. With this information, fault identification modules can use the history of measurements to determine fault magnitude [14, 45]. A unique candidate may not always be attainable if multiple candidates may produce the same effects. In this case, the algorithm must wait until all measurements that should deviate have deviated, or use a time limit.

In MFD, a unique candidate may be obtained, but the next measurement that deviates can always increase the size of the diagnosis. Therefore, ending isolation after a unique candidate is obtained may ignore future evidence of more faults. The most conclusive minimal diagnosis can only be obtained after all measurements have deviated. However, as we have seen, it may be the case that more faults have occurred, but their effects have been completely masked. These candidates would be covered by the minimal diagnosis, but the approach can only be viewed as a *best effort* paradigm. We can only be sure that faults have occurred if we see direct evidence that match only that fault. If faults are assumed to be independent, then we can assign a priori probabilities of individual fault occurrence, and use this to look for a unique *most likely* candidate in the diagnosis. Using probabilities in this way still does not help establish when to terminate isolation, because new measurement deviations can always eliminate previous candidates in favor of larger, less likely candidates. A time limit may be used, or, if combined with quantitative methods, we can wait until the probability of some candidate approaches unity and reaches a threshold probability level, e.g., as in the quantization-based hidden Markov model methods [68, 69]. We will show in Chapter VII how we can determine whether it is possible for a current diagnosis to change if more deviations occur, and thus enable early termination.

Summary

Hypothesis generation produces an initial set of candidates to start monitoring. Prediction provides fault signatures and relative measurement orderings, and progressive monitoring uses these to track fault candidates as new events occur in the system. The algorithms are different depending on the assumptions (i.e., single versus multiple faults), but follow the same basic approach. Because minimality and a fault size limit are used, the isolation algorithms are efficient.

Like the previous TRANSCEND approach, our qualitative fault isolation approach depends heavily on correct symbol generation for correct diagnosis. If an incorrect symbol is provided, then correct

candidates may be eliminated. Therefore tuning the fault detectors and symbol generators is very important in this framework and robust techniques for signal to symbol transformation are necessary [47]. The use of statistical tests helps make the symbol generation step robust. This is critical to avoid generating incorrect fault candidates and eliminating correct candidates.

Fault detection is also crucial in using measurement orderings. If the fault detector for one measurement is more sensitive than another, then it will have a faster response. This may provide inaccurate ordering information to the fault isolation algorithms, which may result in correct candidates being eliminated. In practice, it is desirable to tune the fault detectors to similar sensitivity levels. We do this by using similar parameter values for all fault detectors.

As described, our approach to multiple fault diagnosis uses ideas from consistency-based methods applied to static systems [2, 3]. We focus on dynamic systems, so we use information on fault transients to form conflicts and refine diagnoses. Unlike previous work, our methodology also incorporates temporal information to increase the discriminatory power of the measurements. Other aspects of our approach, such as using candidate cardinality limits, has been used in other work, such as the signed digraph approach in [122]. Consideration of smaller combinations of faults before larger combinations to reduce computational complexity is also used by [122], and has been used in our earlier work in [38]. Other signed digraph approaches also investigate multiple faults [123], however most signed digraphs do not account for the effects of integration and time delays. Other methods for multiple faults use fault propagation graphs which abstract all diagnostic information to tests and alarms, which rely heavily on properly designed tests but do not address the test design problem [124–127]. Our use of incremental consistency checking is also similar to sequential testing approaches [128, 129].

One of the main contributions of this chapter is the introduction of relative measurement orderings into qualitative diagnosis. We combined measurement orderings with fault signatures to form an event-based framework for diagnosis, based on fault traces, fault languages, and fault models. We also described how to check consistency of candidates and form new diagnoses based on the event-based information. In the next chapter, we describe how this work extends to hybrid systems and isolation of both parametric and discrete faults, based on the Hybrid TRANSCEND methodology [14].

CHAPTER VI

QUALITATIVE FAULT ISOLATION IN HYBRID SYSTEMS

This chapter extends the qualitative fault isolation schemes to hybrid systems, where the continuous system behavior may be interspersed with discrete configuration (mode) changes in the system. Fault isolation is still based on observing deviations from nominal behavior in the measurements, however, the algorithms must now deal with mode changes. This chapter builds up the theory for fault isolation in hybrid systems based on measurement deviation events, and describes the associated algorithms for fault isolation. The results of this chapter extend the Hybrid TRANSCEND methodology described in [14, 22].

In hybrid systems, mode changes may occur between fault occurrence and detection, and also after fault detection. Some of these mode changes may be controlled, i.e., based on known controller output, but some may also be autonomous, i.e., the configuration change is a function of the system variables or unobservable fault events. Nominal autonomous mode transitions cannot be predicted after fault occurrence because a correct model of the system is no longer known [14]. This complicates the diagnosis task, and, therefore, requires reasoning about possible mode changes during both hypothesis generation and candidate tracking. This reasoning is performed in Hybrid TRANSCEND [14]. We adopt the Hybrid TRANSCEND framework, and extend the diagnosis scheme to include discrete faults attributed to switching elements.

The diagnoser architecture for hybrid systems is shown in Fig. 37. Given an initial measurement deviation event, σ_0, the diagnoser produces an initial diagnosis, i.e., set of fault hypotheses, d_0, which includes candidates for different possible modes of fault occurrence. The initial diagnosis triggers prediction, which generates the initial set of predictions, P_0, i.e., fault signatures and relative measurement orderings, for all the candidates in d_0 for their hypothesized modes. Tracking further measurement deviations and controlled mode change events updates the candidate set. Measurement deviations may require hypothesizing possible autonomous mode changes, Q_i', which are passed to the prediction module to produce new predictions, P_i, when the mode changes or when multiple fault hypotheses are considered. When controlled mode changes, σ_q, occur, which are known, new predictions are also necessary. After each new event, candidate tracking produces an intermediate diagnosis, d_i. A user-specified stopping criterion terminates isolation and provides the final diagnosis, d.

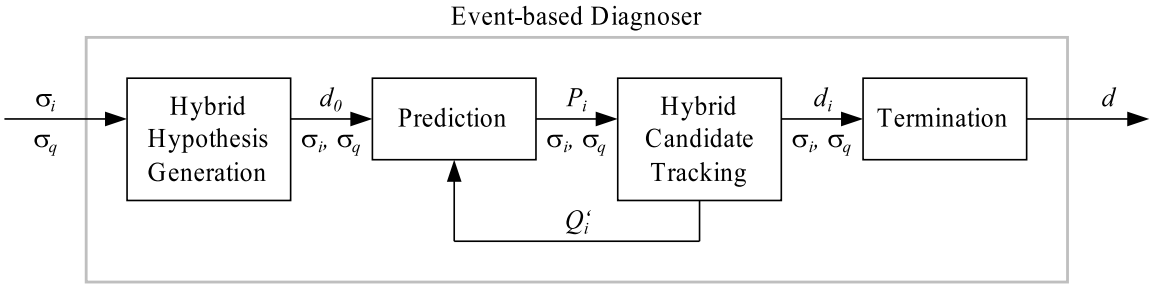


Figure 37: Event-based diagnoser architecture for hybrid systems.

The previous Hybrid TRANSCEND approach assumed only single, abrupt, parametric faults, and did not utilize the temporal order of the measurement deviations in the fault isolation task. We incorporate measurement orderings into the approach, and show how the extended form of the temporal causal graph can be used for integrated parametric, sensor, and discrete fault isolation [36, 37]. Further, we relax the single fault assumption and allow multiple faults to occur, and provide updated algorithms for multiple fault isolation in hybrid systems. The results provided in this chapter are used in diagnoser design for hybrid systems, described in Chapter VII.

The chapter is organized as follows. We first describe the hypothesis generation procedure. Then, we describe how we extend event-based fault modeling to hybrid systems. We next discuss how fault isolation progresses as more events are provided, and how it may terminate. We conclude with a summary of the approach and comparisons to related approaches.

Hybrid Hypothesis Generation

When an initial measurement occurs, generating a + or − symbol representing whether the signal residual increased or decreased, respectively, the hybrid hypothesis generation uses the symbol and hypothesized system mode at the time of fault detection, $\hat{q}(t_d)$, to produce a consistent set of initial fault candidates. We extend hypothesis generation in Hybrid TRANSCEND by hypothesizing the occurrence of discrete faults.

Determining Possible Modes of Fault Occurrence

Because there is a delay associated with fault detection, in many cases faults will not be detected at exactly the point of fault occurrence. Due to this delay, mode changes may occur between the actual time point of fault occurrence and its detection. This requires that we hypothesize the true mode of fault occurrence in order to produce consistent candidates and form the initial diagnosis [14]. For example, if the previously known mode was that a particular switch was already off, then we should

not generate a stuck-off fault of the switch as a candidate. In the following, we use the Hybrid TRANSCEND approach presented in [14] to determine the true mode of fault occurrence, but include discrete fault modes.

Immediately before the time of fault occurrence, t_f, the system was operating nominally in some mode $q(t_f^-)$. The fault, however, is only detected at $t_d \geq t_f$, when the system is in some mode $q(t_d)$. Controlled and autonomous mode changes may have occurred during the interval $[t_f, t_d]$, so it may be the case that $q(t_f^-) \neq q(t_d)$. If a discrete fault occurred, then, since discrete faults induce mode changes in the system, $q(t_f^-) \neq q(t_f^+)$. For parametric faults, though, the mode immediately before and immediately after the fault occurrence is the same, i.e., $q(t_f^-) = q(t_f^+)$.

At the time of fault detection, the system behavior has possibly progressed to some new mode, but the mode predicted by the hybrid observer may be different. Since we no longer have an accurate estimate of the system state when a fault occurs, we actually obtain a set of possible modes of fault occurrence, $\hat{Q}_{t_f^-}$. In order to simplify this problem, we assume that we can accurately track the system mode under nominal conditions [14].

Assumption 9 (Accurate Mode Tracking Under Nominal Conditions)**.** For $t < t_f$, $\hat{q}(t) = q(t)$.

Recall that we assume the observer-estimated mode trajectory is no more than n_{back} mode changes from the mode immediately preceding fault occurrence, i.e., $q(t_f^-)$ (Assumption 2). Then, given Assumptions 2 and 9, we can guarantee the true mode of fault occurrence, $q(t_f^-)$, belongs to our observer-estimated mode trajectory, and is in $\{\hat{q}_{i-n_{back}}, \hat{q}_{i-n_{back}+1}, \ldots, \hat{q}_i - 1, \hat{q}_i\}$, where $\hat{q}_i = \hat{q}(t_d)$. We define the set of possible modes of fault occurrence as $\hat{Q}_{t_f^-}$. Fig. 38, shows how this works. Given $n_{back} = 2$, the possible modes of fault occurrence are given by the set $\hat{Q}_{t_f^-}$. If modes are not changed at high frequency, n_{back} is typically set to 2 [14]. Any number of unpredicted mode changes may have occurred from the true $q(t_f^-)$, but these will be hypothesized in a forward reasoning process, described later.

The `RollBack` procedure performs a backward mode search starting from $\hat{q}(t_d)$ to produce the set of possible modes, $\hat{Q}_{t_f^-}$, in which the fault may have occurred [14]. Since we assume discontinuities are detected at the point of fault occurrence (Assumption 6), then, if a discontinuity is observed, $\hat{Q}_{t_f^-} = \{\hat{q}(t_d)\}$. Since abrupt parametric faults and discrete faults always produce discontinuities on at least one system variable, it is useful to measure these variables because it pinpoints exactly $q(t_f^-)$. For example, an abrupt capacitance fault will produce a discontinuity on the voltage across the capacitor, and a switch turning on or off will produce a discontinuity in the value of the current through the switch.

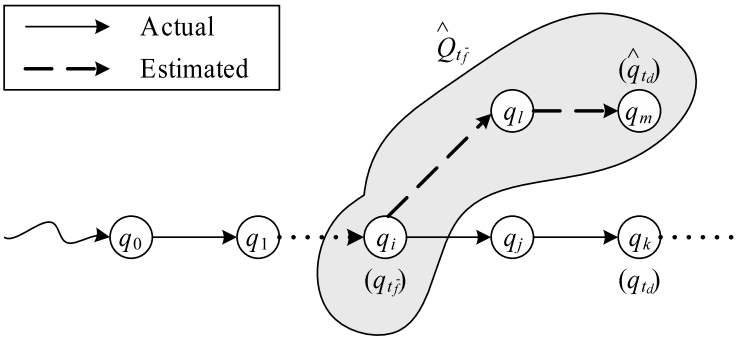


Figure 38: Computing the true mode of fault occurrence.

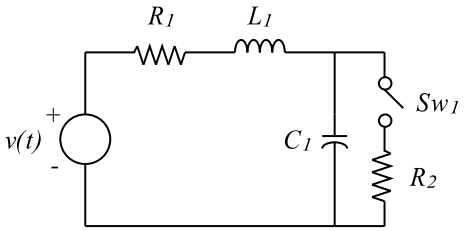
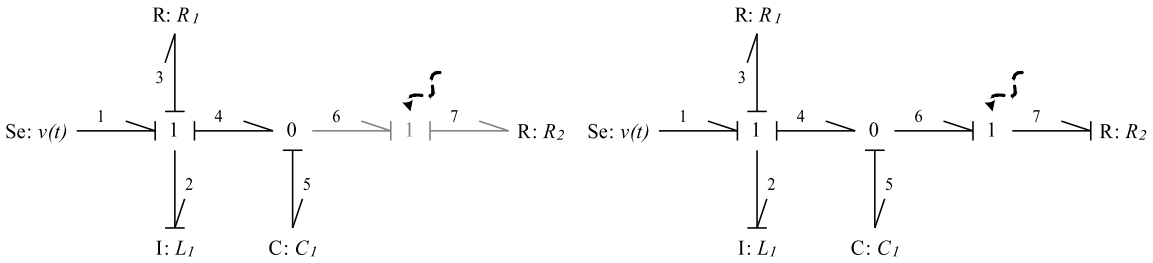


Figure 39: Switched circuit.



(a) Causal HBG when the switch is off.

(b) Causal HBG when the switch is on.

Figure 40: Causal HBGs for the switched circuit.

Example. Consider the circuit given in Fig. 39, which will be used throughout the chapter. The corresponding causal HBGs for the different modes are shown in Fig. 40. We take $Q = \{q_0, q_1, q_{\tau_0}, q_{\tau_1}\}$, representing the nominal modes q_0 for the switch off and q_1 for the switch on, and the faulty modes, q_{τ_0} for the stuck-off fault, τ_0, and q_{τ_1} for the stuck-on fault, τ_1. Consider that a fault is detected when $\hat{q}(t_d) = q_1$. If the mode change to q_1 happened recently, then the fault may have occurred before the mode event when the system was still in q_0. In this case, $\hat{Q}_{t_f^-} = \{q_0, q_1\}$, from which either τ_1 or τ_0 may occur. But, if the deviation was detected as a discontinuity, then $\hat{Q}_{t_f^-} = \{q_1\}$, from which only τ_0 may occur.

We need to derive consistent fault candidates for all potential modes of fault occurrence, subject to the stated assumptions. In order to produce correct candidates, we need to treat autonomous mode changes differently from Hybrid TRANSCEND. As a motivating example, consider a circuit that contains a resistance and a circuit breaker on the same line. Assume a fault that decreases the resistance, causing an increase in the current, which in turn causes the circuit breaker to trip. Because the circuit breaker operation is fast, we may only observe a decrease in the current due to the circuit breaker tripping, rather than the initial increase in current due to the resistance fault. Since the current model of the system is incorrect after fault occurrence, we cannot correctly predict nominal autonomous mode changes [14]. In Hybrid TRANSCEND, however, since only parametric faults are dealt with, it is assumed that the first observed measurement deviation will be from a parametric fault. With this assumption, the `RollBack` procedure in Hybrid TRANSCEND will include the correct mode of fault occurrence, i.e., that where the circuit breaker is on. However, for the actual fault, i.e., the resistance decrease, the current should increase, but for this situation a decrease is observed. Therefore, the correct candidate will not be identified. Even if the mode where the circuit breaker is off is considered, the prediction of the correct fault is no effect on the current, because in that mode the circuit breaker determines the current (i.e., it is set to zero). Still, the correct candidate will not be identified.

To address this problem, we need to link unpredicted nominal autonomous mode changes back to the measurement deviations that they may produce. In our approach, we treat unpredicted, but nominal, autonomous mode changes after a fault occurrence the same as we treat discrete faults (unpredicted, but faulty, autonomous mode changes), because both produce deviations from predicted nominal behavior. As a result, when we generate hypotheses, we determine what can happen *from* the current expected mode of fault occurrence, which includes parametric faults, discrete faults, *and* unpredicted nominal autonomous mode changes. So, we include similar "fault" events for both unpredicted nominal autonomous mode changes and discrete faults, assuming that the autonomous mode changes are of the form where their effects can be predicted (like discrete faults). As a result, in the previous example, we will generate as a hypothesis that the circuit breaker has tripped, because we can predict that it would have caused the decrease in current. Since we saw a deviation from normal behavior, we would have to conclude that some fault, for which we have not yet seen any evidence, has occurred that in turn caused the circuit breaker to trip. Knowing the CSPEC

of the circuit breaker allows us to determine how the variables it depends on may have changed to cause the mode change, which can be mapped back to faults using backward propagation.

So, we need to link the initial measurement deviations back to possible parametric faults, discrete faults, or nominal, but unpredicted, autonomous mode changes. Given a measurement deviation, σ, defined by a measurement m and sign of deviation s, and a TCG for a mode $(V, E, L, l_E)_q$, the hypothesis derivation algorithm computes the possible faults that may have occurred in that mode. `PropagateBackward` extends the algorithm given in [20] by implementing steps to determine which unpredicted autonomous mode changes could have caused the observed deviation σ in addition to the steps that determine if parametric faults are consistent with σ. The TCG for each possible $\hat{q}(t_f^-)$ is used, and determines what could have happened *from* that mode, i.e., what are the changes in the current mode that can explain the initial measurement deviation. These may be changes in parameter values, changes in variables (for additive faults), or the occurrence of fault events that correspond to discrete faults or unpredicted nominal autonomous mode changes.

The extended `PropagateBackward` is given as Algorithm 6. Starting from the vertex associated with the deviated measurement, v_m, `PropagateBackward` traverses the graph backward, visiting instantaneous edges before integrating edges. As an edge is handled, the current sign and the edge label are used to determine the sign to propagate to the predecessor variable. The `DetermineSign` function performs this operation, and simply produces a consistent sign. If not yet visited, the predecessor will be added to the V_{pend} queue. If the edge is parametric, `GetCandidate` determines if the parameter change is a possible fault and returns both the parameter and the sign of its change, or returns nothing. If the predecessor variable itself is a candidate (i.e., because it represents an additive fault represented as a source element) it is added as a fault hypothesis. See [20] for further details on implicating parametric faults.

To handle fault events, we need to extend the traditional backward propagation algorithm. If the predecessor vertex is a fault event vertex (where the fault event is given by τ_v), then it is determined whether the event is possible from the known mode (based on the mode transition function μ), and if so, whether it is consistent with the observation. If, according to the TCG, the variable was supposed to go to zero, and it was positive and decreased or negative and increased, or was supposed to go nonzero, and either increased or decreased, then it is added. For the $+1$ and -1 edges, consistency is checked. Recall that the TCG construction assumes variable values are positive, so, for example, if the edge has a $+1$ label and an increase was observed, this is only consistent with the fault event occurring if the variable is positive, and if the variable was negative, then a decrease would have to be observed for it to be consistent. The signs of estimated variables are also required

Algorithm 6 $F_q \leftarrow$ `PropagateBackward`$(\sigma, (V, E, L, l_E)_q)$

```
(m, s) ← σ
V_pend ← {(v_m, s)}
V_visited ← {v_m}
while V_pend ≠ ∅ do
  (v, s) ← PopFront(V_pend)
  for all (v', l, v) ∈ E do
    if v' is not a fault event then
      if v' ∉ V_visited then
        s' ← DetermineSign(v, s, l)
        if l is instantaneous then
          PushBack(V_pend, (v', s'))
        else
          PushFront(V_pend, (v', s'))
        if l is parametric then
          F_q ← F_q ∪ {GetCandidate(v, s, l)}
        if v'^s ∈ F then
          F_q ← F_q ∪ {v'^s}
    else
      if μ(τ_v, q) ≠ ∅ then
        if l = Z ∧ ((s = - ∧ v̂ > 0) ∨ (s = + ∧ v̂ < 0)) then
          F_q ← F_1 ∪ {τ_v}
        else if l = N ∧ v̂ = 0 ∧ (s = + ∨ s = -) then
          F_q ← F_1 ∪ {τ_v}
        else if l = +1 ∧ ((s = + ∧ v̂ > 0) ∨ (s = - ∧ v̂ < 0)) then
          F_q ← F_1 ∪ {τ_v}
        else if l = −1 ∧ ((s = - ∧ v̂ > 0) ∨ (s = + ∧ v̂ < 0)) then
          F_q ← F_1 ∪ {τ_v}
```

for parametric faults, because determining how a parameter increase or decrease affects the edge's destination vertex depends on the sign of the variable associated with that vertex. Estimated values of all variables in the TCG can be computed from the estimated state variables, so these are all available from the observer.

Example. The TCG for the circuit of Fig. 39 for mode q_1 is given in Fig. 41. Consider that a decrease in the measured current through R_2, or i_3^-, is observed when the switch was expected to be closed. In the circuit, we consider $F = \{C_1^+, C_1^-, L_1^+, L_1^-, R_1^+, R_1^-, R_2^+, R_2^-, \tau_0, \tau_1\}$, for faults in the capacitor, inductor, resistances, and switch. Propagating back in the TCG for the nominal mode q_1, the decrease in i_3 can be explained by f_7^- which in turn can be explained by the fault event τ_0, given f_7 is positive. It can also be explained by R_2^+ or e_7^-. Propagating backwards further can link the change in e_7^- to C_1^+, R_1^+, and L_1^+. Thus `PropagateBackward` for q_1 would return $\{C_1^+, L_1^+, R_1^+, R_2^+, \tau_0\}$.

The overall procedure, `GenerateHypotheses`, is shown as Algorithm 7. Given the initial event σ_0, and given the estimated mode at the time of fault detection, $\hat{q}(t_d)$, the algorithm firsts rolls back to generate possible modes of fault occurrence. For each of these modes, it generates fault hypotheses

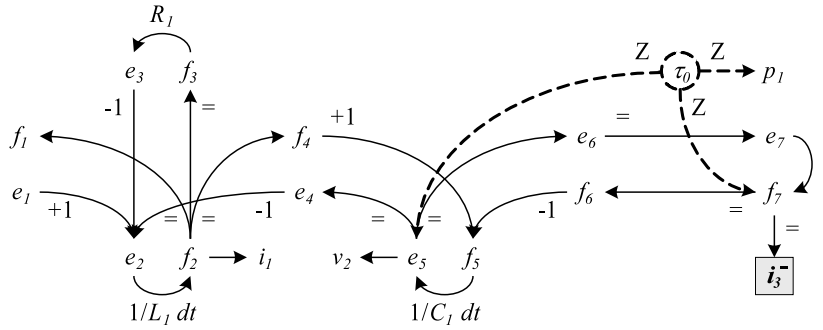


Figure 41: TCG for `PropagateBackward` example.

Algorithm 7 $d_0 \leftarrow$ `GenerateHypotheses`$(\sigma_0, \hat{q}(t_d))$

```
d_0 ← ∅
Q̂_{t_f^-} ← RollBack(q̂(t_d))
for all q̂ ∈ Q̂_{t_f^-} do
    F_q ← PropagateBackward(σ_0, TCG_q̂)
    for all f ∈ F_q do
        d_0 ← d_0 ∪ {(f, μ(f, q(t_f^-)))}
return d_0
```

using `PropagateBackward`. It then forms candidates out of these (i.e., fault, mode pairs) and adds these to d_0, which becomes the initial diagnosis. For discrete faults, it provides the mode induced by the discrete fault, as determined by the mode transition function μ. In other words, it determines $\hat{q}(t_f^+)$ from $\hat{q}(t_f^-)$ given the fault event. If the fault is parametric, it will not change the expected mode.

Example. Consider again that i_3^- is observed and the mode is known to be q_1. Since there are no autonomous transitions that may occur other than discrete faults, the only possible $\hat{q}(t_f^-) \in \hat{Q}_{t_f}$ is that where the switch is on, i.e., q_1. Therefore, the initial diagnosis, d_0, is the set of faults returned by `PropagateBackward` in mode q_1, augmented with the induced fault modes, i.e., $d_0 = \{(C_1^+, q_1),(L_1^+, q_1),(R_1^+, q_1),(R_2^+, q_1),(\tau_0, q_{\tau_0})\}$.

Prediction

In hybrid systems, prediction must account for the hypothesized mode. We extend the prediction algorithm in Hybrid TRANSCEND to generate signatures for unpredicted autonomous mode changes. We also extend our earlier event-based modeling framework to hybrid systems.

As discussed in Chapter IV, mode changes in HBGs explicitly set some variables from nonzero to zero or from zero to nonzero, which forms a special class of discontinuities. We augment fault signatures to include this discrete change information, because it directly indicates unpredicted

autonomous mode changes. We include in the signature additional symbols `N`, `Z`, and `X`, implying zero to nonzero, nonzero to zero, or no discrete change behavior in the measurement from the estimate. Given a measurement m, deviation d, and discrete change dc, we write a signature as an event using $m^{d,dc}$, e.g. $m_1^{+-,X}$. The extended definition of a fault signature is as follows.

Definition 30 ((Extended) Fault Signature)**.** A *fault signature* for a fault f and measurement m in mode q is the qualitative magnitude, slope, and discrete change in m caused by the occurrence of f, and is denoted by $\sigma_{f,m,q} \in \Sigma_{f,m,q}$. We denote the set of all fault signatures for fault f and measurements M in mode q as $\Sigma_{f,M,q}$, where $\Sigma_{f,M,q} = \bigcup_{m \in M} \Sigma_{f,m,q}$.

In hybrid systems, relative measurement orderings are now defined with respect to a particular mode, i.e., $\Omega_{f,M,q}$ represents the orderings for fault f over measurements M in mode q. The notions of fault traces, fault languages, and fault models extend similarly. Therefore, for each fault f in each mode q we have a fault language $L_{f,M,q}$ and a fault model $\mathcal{L}_{f,M,q}$. However, we also need to generate fault models for discrete faults, and to do this, we need to update the `PropagateForward` algorithm. Note that `GenerateHypotheses` (Algorithm 7) applies the mode change induced by the fault. This is especially important for discrete faults, because this is the correct mode in which to make predictions. So in prediction, the goal is to find out the effects of a discrete fault *in* this mode.

For a given mode q, the prediction algorithm computes fault signatures and relative measurement orderings given a parametric or discrete fault f, the set of measurements M, and a TCG, $(V, E, L, l_E)_q$. The extension to the algorithm comes in `UpdateSignature`, shown as Function 8. For each fault, the normal increase/decrease propagation needs to occur in addition to the propagation of the discrete change symbol. The increase/decrease propagation works as in [20], but for discrete faults,the increase or decrease propagation depends on whether the fault causes the junction to turn on or off. If the fault results in a junction turning off, then a decrease is propagated if the junction variable was last known to be positive, otherwise an increase is propagated. The estimated variable sign is needed for parametric faults as well, because an increase or decrease in a parameter value will have a different effect if the immediately affected variable is positive or negative. If the fault results in a junction turning on, then, in general, we do not know whether the junction variables will become positive or negative. However, this is determined during `PropagateBackward`, as shown in the following example.

Example. Consider the τ_1 fault for the circuit. The TCGs for the different modes are shown in Fig. 42. Consider that we observe v_2^+ in mode q_0. When we propagate backward (Fig. 42a), we map v_2^+ to f_6^+, which in turn implicates the τ_1 fault. So in this case, if τ_1 is the actual fault, it will cause

Function 8 $\sigma_{v'} \leftarrow$ `UpdateSignature`$(v, \sigma_v, l, v', \sigma'_v)$

assign consistent signature to $\sigma_{v'}$
if $l = Z$ **then**
  $\sigma_{v'}(dc) =$ `Z`
else if $l = N$ **then**
  $\sigma_{v'}(dc) =$ `N`
else if $\sigma_v(dc) =$ `Z` $\wedge\, l$ is equality or of θ or $1/\theta$ form **then**
  $\sigma_{v'}(dc) =$ `Z`
else if $\sigma_v(dc) =$ `N` $\wedge\, l$ is equality or of θ or $1/\theta$ form **then**
  $\sigma_{v'}(dc) =$ `N`
else
  $\sigma_{v'}(dc) =$ `X`

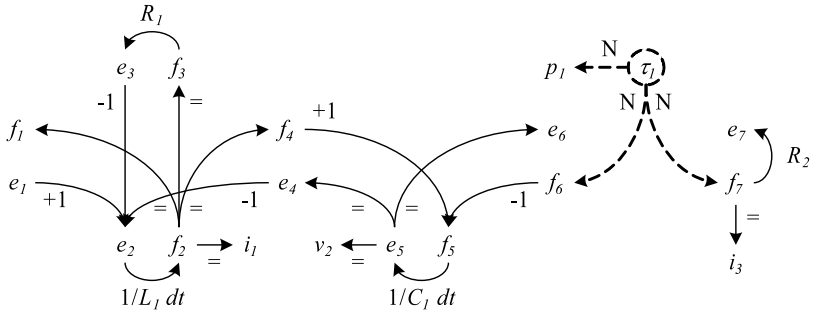
f_6^+. Since the flows at a 1-junction are equal, the `+` change should be propagated forward to both f_6 and f_7 in the TCG for the fault mode, q_{τ_1} (Fig. 42d). Since we assume a consistent direction of change for the effort and flow of the determining bond (Assumption 8), f_7 and e_7 should change in the same direction, so we also know what to propagate to e_7.

For propagation of the discrete change value of the signature, which is denoted by dc in the `UpdateSignatures` function, parametric faults and vertices with the `X` value (no discrete change behavior) initially propagate the `X`. A `Z` symbol (variable goes to from nonzero to zero) will be propagated when an edge with a label of Z is traversed, and will continue propagating along paths of the θ, $1/\theta$, and $=$ form. Similarly, an `N` symbol (variable goes from zero to nonzero) will be propagated when an edge with a label of N is traversed, and will continue propagating along paths of the θ, $1/\theta$, and $=$ form. When an edge that is not of that form is encountered, `X` propagates along the rest of the paths.

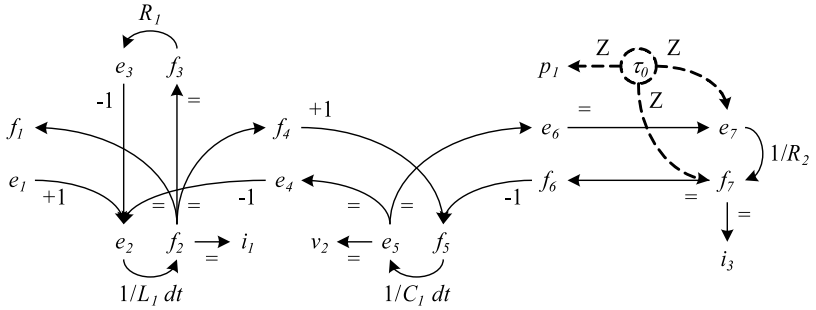
The signatures and orderings for the circuit of Fig. 39 are shown in Table 5. In the circuit, we define the measurements as $M = \{i_1, v_2, i_3\}$, or the current through L_1, the voltage across C_2, and the current through the switch Sw_1, respectively. In the HBG, these correspond to f_2, the flow of bond 2, e_5, the effort of bond 5, and f_7, the flow of bond 7. In deriving the signatures, we have assumed that variable values are nominally positive.

Example. Consider the τ_0 fault. The fault-induced mode, $\hat{q}(t_f^+) = q_{\tau_0}$, is given by the TCG in Fig. 26a. Propagation begins at the fault event τ_0. This results in an immediate change in p_1, resulting in a signature of `(-0,Z)`, and an immediate decrease in f_7, assuming f_7 was positive, resulting in `(-*,Z)` for i_3[1]. It will also cause a decrease in f_6, which will propagate to the rest of the system. The `Z` symbol is also propagated to the immediate variables. This propagation stops at f_5, because f_5 is $f_4 - f_6$. The residual for v_2 will exhibit a first-order increase due to the integration

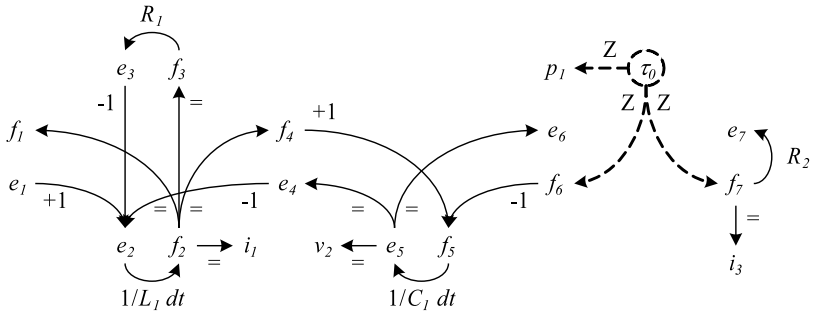
[1]Even though the TCG does not predict the `*`, it is necessary to include. The slope in the new mode does not change, but the slope in the estimated mode may change, so the residual may have a nonzero slope.



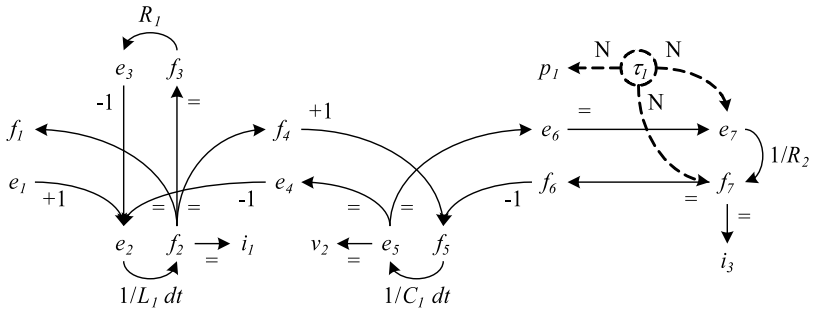
(a) TCG for mode q_0.



(b) TCG for mode q_1.



(c) TCG for mode q_{τ_0}.



(d) TCG for mode q_{τ_1}.

Figure 42: TCGs for the nominal and fault modes.

along the edge labeled with $1/C_1 dt$, resulting in a signature of `(0+,X)`. The increase propagates to

Table 5: Fault Signatures and Relative Measurement Orderings for the Switched Circuit

Candidate	i_1	v_2	i_3	Measurement Orderings
(C_1^+, q_0)	`0+,X`	`-+,X`	`00,X`	$v_2 \prec i_1, i_1 \prec i_3, v_2 \prec i_3$
(C_1^-, q_0)	`0-,X`	`+-,X`	`00,X`	$v_2 \prec i_1, i_1 \prec i_3, v_2 \prec i_3$
(L_1^+, q_0)	`-+,X`	`0-,X`	`00,X`	$i_1 \prec v_2, i_1 \prec i_3, v_2 \prec i_3$
(L_1^-, q_0)	`+-,X`	`0+,X`	`00,X`	$i_1 \prec v_2, i_1 \prec i_3, v_2 \prec i_3$
(R_1^+, q_0)	`0-,X`	`0-,X`	`00,X`	$i_1 \prec v_2, i_1 \prec i_3, v_2 \prec i_3$
(R_1^-, q_0)	`0+,X`	`0+,X`	`00,X`	$i_1 \prec v_2, i_1 \prec i_3, v_2 \prec i_3$
(R_2^+, q_0)	`00,X`	`00,X`	`00,X`	$\varnothing$
(R_2^-, q_0)	`00,X`	`00,X`	`00,X`	$\varnothing$
(C_1^+, q_1)	`0+,X`	`-+,X`	`-+,X`	$v_2 \prec i_1, i_3 \prec i_1$
(C_1^-, q_1)	`0-,X`	`+-,X`	`+-,X`	$v_2 \prec i_1, i_3 \prec i_1$
(L_1^+, q_1)	`-+,X`	`0-,X`	`0-,X`	$i_1 \prec v_2, i_1 \prec i_3$
(L_1^-, q_1)	`+-,X`	`0+,X`	`0+,X`	$i_1 \prec v_2, i_1 \prec i_3$
(R_1^+, q_1)	`0-,X`	`0-,X`	`0-,X`	$i_1 \prec v_2, i_1 \prec i_3$
(R_1^-, q_1)	`0+,X`	`0+,X`	`0+,X`	$i_1 \prec v_2, i_1 \prec i_3$
(R_2^+, q_1)	`0-,X`	`0+,X`	`-+,X`	$v_2 \prec i_1, i_3 \prec i_1, i_3 \prec v_2$
(R_2^-, q_1)	`0+,X`	`0-,X`	`+-,X`	$v_2 \prec i_1, i_3 \prec i_1, i_3 \prec v_2$
(τ_0, q_{τ_0})	`0-,X`	`0+,X`	`-*,Z`	$v_2 \prec i_1, i_3 \prec i_1, i_3 \prec v_2$
(τ_1, q_{τ_1})	`0+,X`	`0-,X`	`+-,N`	$v_2 \prec i_1, i_3 \prec i_1, i_3 \prec v_2$

i_1 and changes sign, providing a second-order decrease (due to $1/L_1 dt$) at i_1, yielding `(0-,X)`. Based on the fault path analysis, a deviation will appear first in i_3, followed by v_2, followed by i_1.

Based on the updated `PropagateForward` algorithm, we can derive signatures and orderings for faults in any given mode, and this allows us to create our fault models. Fig. 43 shows the resultant fault models for the circuit example for the nominal modes q_0 and q_1, and the fault-induced modes modes q_{τ_0} and q_{τ_1}. Note that we also require fault models for parametric faults in the fault modes, e.g., $\mathcal{L}_{R_1^+, M, q_{\tau_0}}$, but these are omitted from the figure because they are similar to those for the nominal modes, e.g., $\mathcal{L}_{R_1^+, M, q_{\tau_0}}$ captures the same traces as $\mathcal{L}_{R_1^+, M, q_0}$. The observed effects of discrete faults depend on the expected mode of the system, therefore, without loss of generality, we assume fault events are specific to the originating mode.

Example. Take R_1^+ in q_1. Its orderings specify that i_1 must deviate before v_2 and i_3. Therefore, i_1^{0-} is first, followed by v_2^{0-} and i_3^{0-} in either order. In mode q_0, i_3 is not affected, so $\mathcal{L}_{R_1^+, q_1}$ has no event for i_3. Since a fault in R_2 produces no observable effects in q_0, its fault models have no events.

Hybrid Candidate Tracking

Hypothesis generation produces the set of fault candidates consistent with the initial measurement deviation, and prediction computes future effects these candidates will have on the measurements. In hybrid systems, candidate tracking uses the predictions (i.e., signatures and orderings, or, equivalently, fault models) to match candidates to new measurement deviations, but must also refine

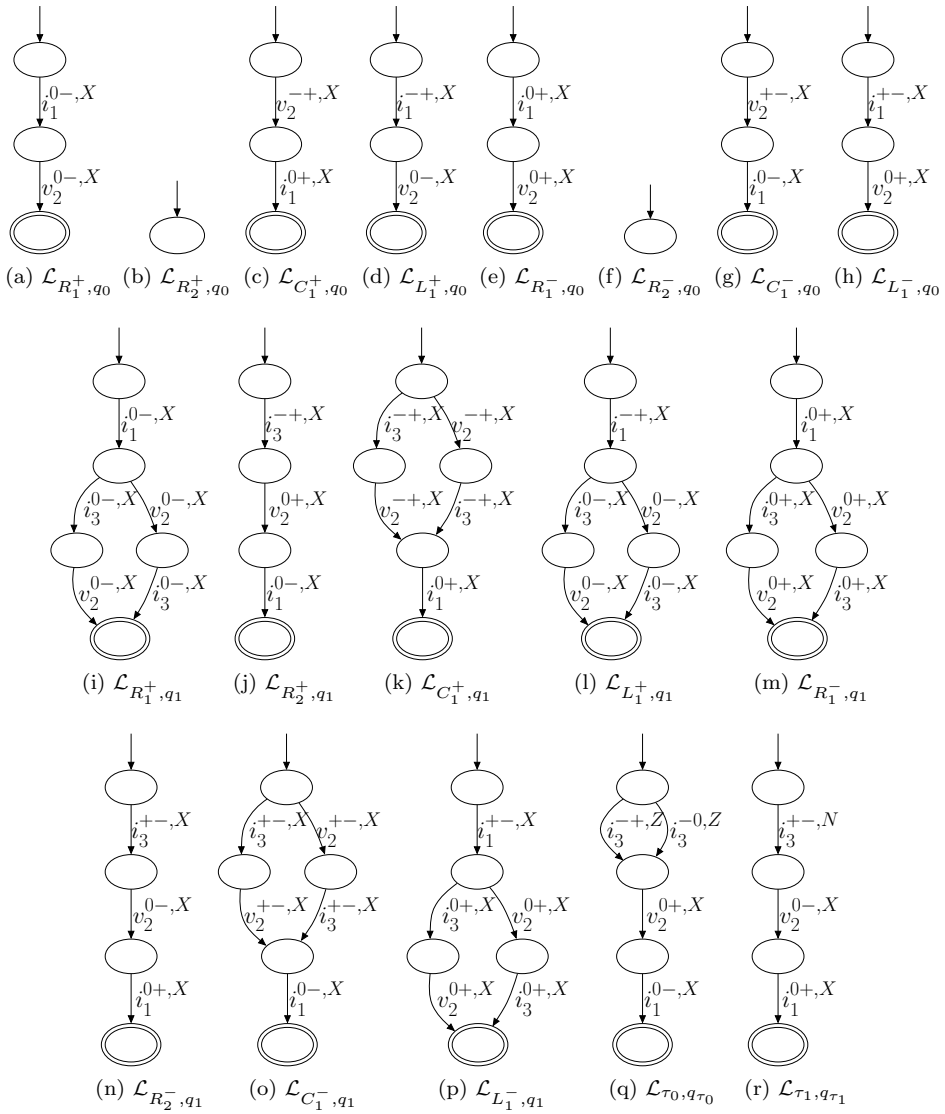


Figure 43: Fault models for the faults of the circuit.

candidates and their predicted effects when mode changes occur. Measurement deviations may indicate new faults or unpredicted autonomous mode changes. We extend the candidate tracking algorithms of Hybrid TRANSCEND to use both fault signatures and relative measurement orderings, to isolate both parametric and discrete faults, and to isolate multiple faults.

In hybrid systems, mode changes may occur after fault occurrence that affect the diagnosis. These mode changes may be controlled or autonomous. With controlled mode changes, we can produce a new nominal reference from which symbol generation calculates measurement deviations. The occurrence of autonomous mode changes (such as discrete faults) cannot be predicted correctly after a fault because a correct model of the system is no longer known [14]. Thus, autonomous mode

changes will produce deviations from *expected* nominal behavior which must be accounted for during candidate tracking, otherwise incorrect results may be obtained. As part of the tracking, we need to hypothesize the occurrence of autonomous mode changes that are attributed to additional discrete faults as well as nominal autonomous mode changes that are consistent with the observed behavior.

In Hybrid TRANSCEND, when a measurement deviation event is observed, one of two things happens. Either a candidate is consistent with the deviation, or it is inconsistent. For example, consider a circuit with a circuit breaker. Assume that a fault occurs which causes some measurement deviations that are detected and therefore the fault is hypothesized, but also which causes an increase in the current through the circuit breaker, which in turn causes the circuit breaker to trip. Assume that the increase due to the fault was not detected before the circuit breaker tripped. Then, the measurement deviation observed with respect to the expected mode will be a decrease in that current. The prediction for the fault is an increase in the current, but since a decrease was observed, an inconsistency is present. If inconsistent, Hybrid TRANSCEND handles this by hypothesizing possible autonomous mode changes using a `RollForward` process, which determines which possible autonomous mode changes may have occurred [14]. If mode changes cause the discrete change behavior, i.e., generate `N` and `Z` symbols in the measured variables, then this process can be made more efficient.

In Hybrid TRANSCEND, once an autonomous mode change is hypothesized, a new parallel thread is created which must update its nominal reference with the mode change, in order for future symbol generation to be correct with respect to the new reference. This results in multiple threads for each hypothesized autonomous mode change, where each has a new model and new symbol generators that use the updated reference for the candidate. In this dissertation, we simplify the approach by keeping only a single reference, that of the expected mode at the point of fault occurrence, which changes only with controlled mode change events. Therefore, parametric faults, discrete faults, and unpredicted nominal autonomous mode changes all produce deviations from this reference. With a single reference, we have to treat autonomous mode changes as faults, i.e., they are placed in F, because they represent some change from the expected behavior. Since the effects of parametric faults and unpredicted mode changes may become interleaved due to masking and relative times of occurrence, we must treat the problem as a multiple fault problem. So, we deal with multiple fault hypotheses, which include parametric faults, unpredicted nominal autonomous mode changes, and discrete faults. This assumes a class of autonomous mode changes that act as discrete faults, i.e., their effects on the measurements can be predicted using the discrete fault methodology, such as a

circuit breaker tripping. As we will show later, using the `N` and `Z` symbols, and measuring variables for which they are produced, makes the approach very powerful for the type of systems we deal with.

In addition to being much more computationally efficient, the single-reference approach is suitable for a practical class of physical systems, such as the electrical distribution system presented in Chapter IX, where mode changes occur from relays and circuit breakers (i.e., state variables are not reset due to mode changes). Although the single-reference approach is more efficient, since we do not separate out the effects of autonomous mode changes and actual faults, we may lose diagnostic power, because the effects due to mode changes can mask those of faults. In the following, we describe our approach using the single-reference paradigm, and leave its extension to the multi-reference paradigm as future work.

Refining Hybrid System Hypotheses

As with continuous systems, we need to define the candidate language for a hybrid candidate in order to formally characterize consistency of candidates. As discussed, traces for hybrid candidates contain both controlled mode change events and measurement deviation events. We denote the set of possible measurement deviation events as Σ_M. Given modes $Q = \{q_0, q_1, \ldots, q_r\}$, we abstract the mode change events to $\Sigma_Q = \{\sigma_{q_0}, \sigma_{q_1}, \ldots, \sigma_{q_r}\}$, where σ_{q_i} indicates an event that changes the system to mode q_i.

In defining the language, controlled mode changes are easy to deal with. They produce a new nominal reference for symbol generation, and all hypothesized faults must be consistent with the predictions in the new mode for new measurements that deviate. In other words, for any future measurement deviations, a candidate must be consistent with the predictions for the new mode, with previously deviated measurements projected out.

Example. Assume R_2^- occurs in mode q_1, followed by a controlled mode change to q_0, as shown in Fig. 44. Here, the trace is $i_3^{+-,X} v_2^{0-,X} \sigma_{q_0}$. When changing to q_0, the nominal reference changed, so since i_1 did not deviate enough to cross the threshold before the mode change, it will not do so in the new mode, because the fault does not continue to affect the measurement.

Unpredicted autonomous mode changes also affect the diagnosis. We need to also predict what unpredicted mode changes may have occurred from the expected mode in order to generate correct diagnoses.

Example. Assume that R_2^- and τ_0 both occur, as shown in Fig. 45. Note that if τ_0 is an unpredicted nominal autonomous mode change, instead of a discrete fault, we treat it the same way. The observed

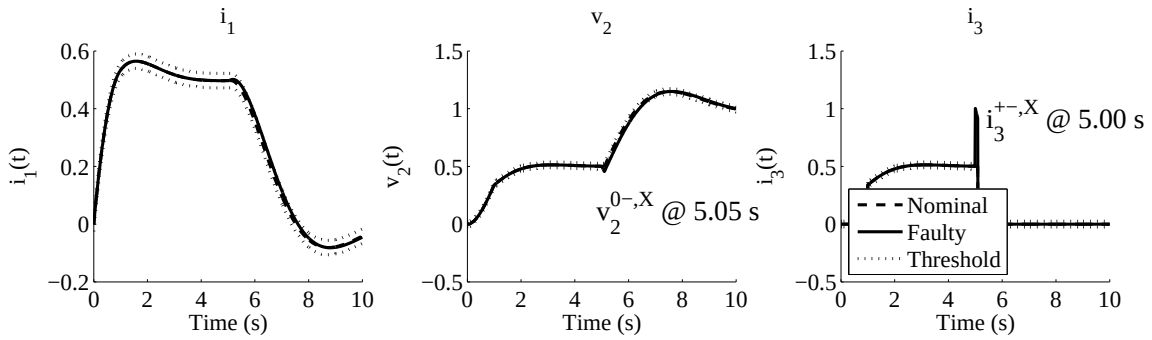


Figure 44: Faulty versus nominal behavior for the circuit measurements. R_2^- (50% decrease) is injected at 5.00 s and a controlled mode change to q_0 at 5.10 s.

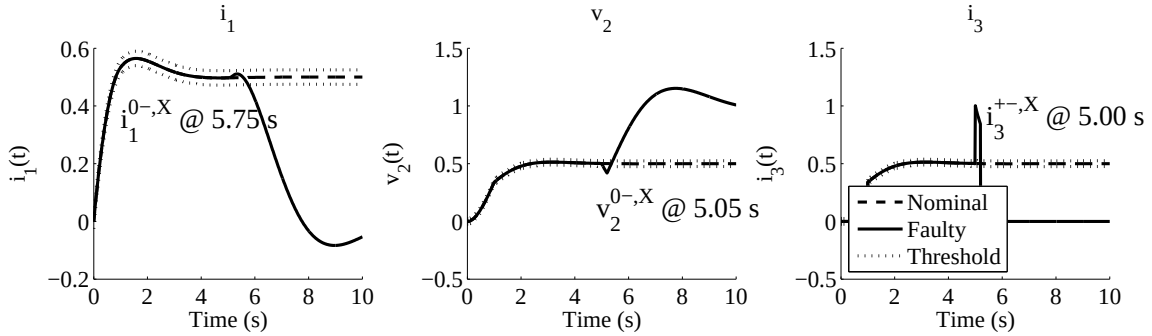


Figure 45: Faulty versus nominal behavior for the circuit measurements. R_2^- (50% decrease) is injected at 5.00 s and τ_0 is injected at 5.20 s.

trace is $i_3^{+-,X} v_2^{0-,X} i_1^{0-,X}$. This trace is not consistent with any single fault in the expected mode, q_1. Therefore, multiple changes to the system must have occurred. We can see that $i_3^{+-,X} v_2^{0-,X}$ is consistent with R_2^- in q_1, and that $i_1^{0-,X}$ is consistent with τ_0 then occurring. So, using a similar approach to MFD in continuous systems, we can obtain a candidate with both R_2^- and τ_0.

From the example, it is clear that the same principles from multiple fault approach in continuous systems apply here, given an expected mode. To determine what new faults are consistent, we check with respect to measurements that have not yet deviated. There may be different possible modes of fault occurrence, depending on the history of control actions, therefore, the set of consistent faults depend also on the expected mode of fault occurrence. For example, in the circuit, if the mode is known to be q_1 immediately before fault occurrence, then the stuck-on fault, τ_1, should not be considered as a fault candidate, because if it did occur we would not have seen a deviation, because q_1 and q_{τ_1} map to the same mode of the corresponding junction, i.e., the junction is on for both modes.

We can now define a *candidate trace* for hybrid candidates.

Definition 31 ((Hybrid) Candidate Trace)**.** An event trace $\lambda = \sigma$ is a *candidate trace* for $c = (F_i, q_i)$ and initial mode of fault occurrence q_0, if $\sigma \sqsubseteq \lambda' \in L_{f,M,q_i}$ for some $f \in F_i$ where $q_i = \mu(f, q_0)$. An event trace $\lambda = \lambda_i \sigma_{i+1}$ is a *candidate trace* for $c = (F_i, q_{i+1})$ and initial mode of fault occurrence q_0, if λ_i is a candidate trace for (F_i, q_i), and if $\sigma_{i+1} \in \Sigma_Q$ then $\mu(\sigma_{i+1}, q_i) = q_{i+1}$, or if $\sigma_{i+1} \in \Sigma_M$ then $q_i = q_{i+1}$ and $\sigma_{i+1} \sqsubseteq \lambda' \in L_{f,M-M_i,q_{i+1}}$ for some $f \in F_i$. A candidate trace for c with initial mode q_0 is denoted as λ_{c,q_0}.

Clearly, there may be an infinite number of candidate traces because controlled mode changes may keep occurring indefinitely. However, as in continuous systems, we are only concerned with *maximal* traces, i.e., those for which all measurements that will deviate in the current mode have deviated (as with fault traces).

Definition 32 (Maximal (Hybrid) Candidate Trace)**.** A candidate trace λ_{c,q_0} for $c = (F_i, q_i)$ is *maximal* if for all $f \in F_i$, $L_{f,M-M_i,q_i} = \varnothing$, where M_i is the set of deviated measurements for λ_{c,q_0}.

Now, we can define the *language* of a candidate c with respect to an initial mode of fault occurrence q_0, L_{c,M,q_0}, as the set of maximal candidate traces for c starting in q_0.

Definition 33 ((Hybrid) Candidate Language)**.** The *candidate language* for candidate c, measurements M, and initial mode of fault occurrence q_0, denoted as L_{c,M,q_0}, is the set of all maximal candidate traces λ_{c,q_0}.

The candidate language consists of all maximal traces for which at each point in the trace, there is a fault whose abbreviated fault language (that for the current candidate mode with deviated measurements projected out) contains a trace that starts with the new event, if the new event is a measurement deviation. If the new event is a controlled mode change, then to be consistent the candidate mode is changed. In simpler terms, a maximal trace is consistent with a candidate if the mode of the candidate can be reached via the sequence of controlled mode changes in the trace and the autonomous mode changes in the candidate, and the measurement deviations within the trace match the faults in the intermediate modes. As with continuous systems diagnosis, the definition of the candidate language makes it clear that we can check consistency incrementally, in which we project out measurements that have already deviated. We use a (hybrid) hypothesis set formulation, which extends the original hypothesis set by including modes.

Definition 34 ((Hybrid) Hypothesis Set)**.** For a fault set $F = \{f_1, f_2, \ldots, f_n\}$, deviated measurements M_i, modes $Q = \{q_1, q_2, \ldots, q_r\}$, and fault languages $\{L_{f_1,M-M_i,q_1}, \ldots, L_{f_1,M-M_i,q_r},$

$L_{f_2,M-M_i,q_1}, \ldots, L_{f_n,M-M_i,q_r}\}$, the hypothesis set $h_{F,M_i}(\sigma_{i+1})$ for the new event σ_{i+1} is computed as the set of all $(f,q) \in F \times Q$, such that there exists some $\lambda \in L_{f,M-M_i,q}$ where $\sigma_{i+1} \sqsubseteq \lambda$.

Using the extended hypothesis set, diagnosis works similarly to MFD in continuous systems, only candidates also contain the mode, which brings in additional reasoning. When a controlled mode change occurs, all current candidates are updated to the new mode, although if an autonomous mode change is included in the candidate, it may not change to the expected mode. For example, if the candidate has hypothesized that switch is stuck-off, and the controller command is for the switch to turn on, the candidate mode will not change. When a measurement deviation occurs, then if there are inconsistent candidates, we find a hypothesis set, which essentially represents the set of "new" faults that may have occurred since the last measurement deviation, although it may also include faults which are in the current diagnosis. The hypothesis set includes (*i*) new faults in the current mode, (*ii*) unpredicted nominal autonomous mode changes, and (*iii*) discrete faults. The $\wedge$ operator is defined as follows for hybrid systems.

Definition 35 ($\wedge$ operator)**.** For diagnosis d and hypothesis set h, $d \wedge h$ is defined as $\{(F_i \cup \{f_j\}, \mu(f_j, q_i))) : (F_i, q_i) \in d, (f_j, \mu(f_j, q_i)) \in h, \mu(f, q_i) \neq \varnothing\}$.

For each candidate in the current diagnosis and candidate in the hypothesis set, we add to the resultant diagnosis a candidate with a new set of faults formed from the faults of the diagnosis candidate and the fault in the hypothesis set candidate (which includes autonomous mode changes). The mode of the new candidate is determined to be the mode of the hypothesis set candidate, if the mode change is possible (if not, the new candidate is not included). If that candidate is a parametric fault, the mode will not change, but if it is an autonomous mode change, it will.

We define the overall operation using $\wedge_l$.

Definition 36 ($\wedge_l$ operator)**.** For diagnosis d and hypothesis set h, $d \wedge_l h$ is defined as $\{(F_i, q_i) \in d \wedge h : |F_i| \leq l, (\neg\exists (F_j, q_i) \in d \wedge h, F_j \subset F_i), F_i$ satisfies Assumption 4$\}$.

The $\wedge_l$ operator is similar to the corresponding operator for continuous systems, only the mode is considered when pruning candidates to make the diagnosis minimal. According to our definition of minimal diagnoses (Definition 5), we only prune a candidate if a second candidate exists for the same mode that contains a subset of the faults in the first candidate. Like continuous systems, we also remove candidates that violate our candidate cardinality limit l (Assumption 1), and those that violate the persistency assumption (Assumption 4).

Candidate tracking is shown as Algorithm 9. It begins with the initial diagnosis d_0, consisting of single-fault candidates obtained by `GenerateHypotheses` for event σ_0. Since hypothesis generation

Algorithm 9 $d \leftarrow$ `HybridCandidateTracking`(d_0, σ_0)

```
d ← ∅
for all ({f_i}, q_i) ∈ d do
   P_{f_i,q_i} ← PropagateForward(f_i, M, TCG_{q_i})
   if (f_i, q_i) consistent with σ_0 then
      d ← d ∪ {({f_i}, q_0)}
while termination condition is false do
   wait for σ
   if σ is a mode change event then
      d ← UpdateHypotheses(d, σ)
   else if σ is a measurement deviation event then
      d ← RefineHypotheses(d, σ)
return d
```

Algorithm 10 $d \leftarrow$ `UpdateHypotheses`(d, σ)

```
for all (F_i, q_i) ∈ d do
   q_i ← μ(q_i, σ)
   for all f_i ∈ F_i do
      P_{f_i,q_i} ← PropagateForward(f_i, TCG_{q_i})
return d
```

uses only the initial sign of the deviation and does not use measurement orderings in selecting initial candidates, we first predict the effects of the faults. We then check consistency using the full signatures and measurement orderings. If consistent with the predictions, the candidate is placed in d.

After initializing d, `HybridCandidateTracking` waits for events to occur. Because both controlled mode change events and measurement deviation events may occur, two algorithms are used. The first, `UpdateHypotheses`, changes the modes of all the current candidates in response to a controlled mode change event. The second, `RefineHypotheses`, refines the current diagnosis in response to a measurement deviation event. When the termination condition is satisfied, fault isolation stops.

When a controlled mode change event occurs, candidates are updated using Algorithm 10. If the candidate includes an autonomous mode change (such as a discrete fault) then the new candidate mode will not be the mode expected by the mode change event. For example, a turn on command to a stuck-off switch will not actually change the mode. For each candidate, we apply the mode change. The change is also applied in the observer, which in turn produces a new nominal reference for fault detection and symbol generation. Because the effects of the faults may be different in the new mode [14], we need to predict the effects of the fault in the new mode. Therefore, the prediction algorithm, `PropagateForward` (Algorithm 1) is called.

`RefineHypotheses` for hybrid diagnosis is shown as Algorithm 11. If all the candidates in the current diagnosis can explain the new deviation, we retain the previous diagnosis, because it is minimal. If not, we obtain the hypothesis set, which is determined using a combination of

Algorithm 11 $d_{i+1} \leftarrow$ `RefineHypotheses`(d_i, σ_{i+1}) (Hybrid Diagnosis)

```
d_{i+1} ← ∅
d_{i+1} ← ∅
if all c ∈ d_i consistent with λ_i σ_{i+1} then
   d_{i+1} ← d_i
else
   h_{i+1} ← ∅
   for all (., q_i) ∈ d_i do
      F_{i+1} ← PropagateBackward(σ_{i+1}, TCG_{q_i})
      for all f_{i+1} ∈ F_{i+1} do
         q_{i+1} ← μ(f_{i+1}, q_i)
         P_{f_{i+1},M,q_{i+1}} ← PropagateForward(f_{i+1}, M, TCG_{q_{i+1}})
         if (f_{i+1}, q_{i+1}) consistent with σ_{i+1} for M − M_i then
            h_{i+1} ← h_{i+1} ∪ {(f_{i+1}, q_{i+1})}
   d_{i+1} ← d_i ∧_l h_{i+1}
return d_{i+1}
```

`PropagateBackward` and `PropagateForward` to obtain consistent candidates, as in the continuous case, except this is done for all modes of the current diagnosis, because we need to find changes from the current possible modes. Note that the hypothesis set may contain parametric faults, nominal autonomous mode changes, and discrete faults. We then compose the hypothesis set h_{i+1} with the old diagnosis d_i using $\wedge_l$, which takes the logical and, and prunes the resultant diagnosis on a per-mode basis, which is consistent with our definition of minimality (Definition 5). The and operation does not combine two candidates that imply both a mode change and a new fault (except for autonomous mode changes, where the mode change and new fault refer to a single change) because it would not be minimal. Note also that in `RefineHypotheses`, we do not compute the full hypothesis set according to Definition 34, because we only compute parametric faults in the current mode of a candidate, or autonomous mode changes which are plausible from the current mode of a candidate, which is more efficient.

Example. Consider that the initial mode is q_0 with $l = 1$. The first observation is $v_2^{-+,X}$. The initial diagnosis is then $\{(C_1^+, q_0)\}$. The candidate (C_1^+, q_1) is also in the complete hypothesis set, but we do not include it because the mode is known to be q_0, i.e., $\mu(C_1^+, q_0) = q_0$, so anding with $(\varnothing, q_0)$, we still get (C_1^+, q_0), therefore we do not need to compute (C_1^+, q_1) in the hypothesis set. In further examples we provide only the truncated form of the hypothesis set. Under the single fault assumption, fault isolation may stop here, but we continue for the purposes of illustration. The second observation is i_1^{0+}, and $\{(C_1^+, q_0)\}$ is still our diagnosis. Then a controlled mode change to q_1 occurs. Now, future measurement deviations must be consistent with this mode, and the diagnosis becomes $\{(C_1^+, q_1)\}$. The third observation is $i_3^{-+,X}$, and the diagnosis remains $\{(C_1^+, q_1)\}$.

Consider that the initial mode is q_1 with $l = 1$. The first observation is $i_3^{-+,Z}$. Because `GenerateHypotheses` (Algorithm 7) considers possible mode changes from q_1, the initial hypothesis set and diagnosis are both $\{(\tau_0, q_{\tau_0})\}$. Since controlled mode changes have no effect in this mode, the diagnosis will remain $\{(\tau_0, q_{\tau_0})\}$ under the single fault assumption. Without the use of the discrete change feature for the signature, we would also have to consider the candidate (R_2^+, q_1), and would not be able to distinguish it from τ_0.

Consider the fault trace $i_1^{0+,X} v_2^{0-,X} \sigma_{q_1} i_3^{-+,Z}$ with the circuit starting in mode q_0 and $l = 3$, i.e., the initial diagnosis is $\{(\varnothing, q_0)\}$. After $i_1^{0+,X}$, the hypothesis set is $\{(R_1^-, q_0)\}$ and the diagnosis then becomes $\{(R_1^-, q_0)\}$. After $v_2^{0-,X}$, the hypothesis set is $\{(R_1^+, q_0), (L_1^+, q_0)\}$. Note that (R_2^-, q_0) and (τ_1, q_{τ_1}) are not included because they require a deviation in i_3 first. The diagnosis then becomes $\{(R_1^- R_1^+, q_0), (R_1^- L_1^+, q_0)\}$. Pruning, we obtain simply $\{(R_1^- L_1^+, q_0)\}$, because $(R_1^- R_1^+, q_0)$ violates fault persistency (Assumption 4). After a controlled mode change to q_1 from event σ_{q_1}, the diagnosis is updated to $\{(R_1^- L_1^+, q_1)\}$. After $i_3^{+-,N}$, the hypothesis set is $\{(\tau_0, q_{\tau_0})\}$, since τ_0 is possible from q_1. The new diagnosis is then $\{(R_1^- L_1^+ \tau_0, q_{\tau_0})\}$, which implies that three faults have occurred and the mode changed to q_{τ_0}. If τ_0 was instead an unpredicted nominal autonomous mode change, the result would be the same, since we would treat it as a fault.

The approach here is the most general and solves the SFD and MFD problems in either continuous or hybrid systems, given the assumptions made. If we eliminate the modes, it reduces to MFD in continuous systems. As shown with the continuous MFD approach, setting $l = 1$ solves the continuous SFD problem.

Terminating Fault Isolation

Candidate tracking stops when the termination condition is true. In the hybrid case, the mode does not matter, so candidate tracking may terminate not on a unique candidate, but on a unique fault set, e.g., termination would occur on both the candidate $\{(f_1, q_0)\}$ and the candidate $\{(f_1, q_0), (f_1, q_1)\}$. The mode the system moved into is not relevant, all that is needed is what fault occurred. With this information, fault identification modules can use the history of measurements and known mode changes to determine fault magnitude [14, 45].

With more measurement deviations, the candidates may change. The most conservative strategy, then, is to wait for all measurements that will deviate to deviate. For hybrid systems, a mode change may change which measurements are reachable by the fault. Mode changes may be autonomous so it may be unknown when the next will occur. In this case, a time limit may have to be used in

order to ensure isolation stops after a finite amount of time. We will show in Chapter VII how we can determine whether it is possible for a current diagnosis to change if more deviations occur, and thus enable early termination.

Summary

Hypothesis generation produces an initial set of candidates, and this is followed by a tracking procedure to refine the candidates. Prediction provides fault signatures and relative measurement orderings, and the candidate tracking uses these to match new events to consistent candidates. During isolation, unpredicted autonomous mode changes (which may be nominal or faulty) are hypothesized based on the measurement deviations they create. The approach assumes a single reference for symbol generation, but can be extended to the multiple-reference approach used by Hybrid TRANSCEND.

One of the main contributions of this chapter is the extension of our event-based diagnosis framework to hybrid systems. We described how to check consistency of candidates and form new diagnoses based on the event-based information. The other main contribution is the extension of Hybrid TRANSCEND to diagnose discrete faults, for which we take advantage of the discrete behavior they cause in certain variables to improve fault isolation and help distinguish between discrete and parametric faults. We describe in the next chapter how this reasoning can be compiled into event-based diagnosers, and how it establishes notions of diagnosability for a system.

CHAPTER VII

DIAGNOSER DESIGN

Our diagnosis framework establishes an event-based diagnosis paradigm for continuous systems with extensions to hybrid systems. Discrete-event system methods are an important framework for event-driven diagnosis in safety-critical systems, since they comprise a well-developed theory that facilitates both diagnosability analysis and systematic construction of computationally efficient online diagnosers. Existing DES diagnosers are designed as extended observers that estimate the system state under nominal and faulty conditions [5, 6, 62, 63]. Although these methods have been used in many practical diagnosis applications [5, 130–132], they are not suitable for systems with complex continuous dynamics. Quantizing the continuous behavior using a finite set of states and events results in large, nondeterministic models that degrade the performance and increase the computational requirements of the diagnosis algorithms [68, 69, 85]. In the presence of faults, these models become increasingly complex, and deriving such models for different fault magnitudes is computationally inefficient.

In contrast to traditional DES approaches, our approach focuses on constructing DES-style diagnosers for isolating parametric and discrete faults in continuous systems and hybrid systems, based on a *qualitative* abstraction of the measurement deviations from the nominal behavior. In our approach, we essentially compile our qualitative fault isolation algorithms for a given system, based on its fault models, to form an event-based diagnoser. The contributions of the approach center on: (*i*) systematic design of event-based diagnosers, (*ii*) diagnosability analysis of systems, and (*iii*) a spectrum of diagnoser implementations that trade off space and time efficiency. Preliminary results for single fault diagnosis of continuous systems have appeared in [82, 86, 96, 133], and multiple fault diagnosis for continuous systems in [38, 39].

First, we present notions of diagnosability, formalized within our framework. We then describe our event-based diagnosers. Because multiple fault diagnosis builds on single fault diagnosis, and hybrid systems diagnosis builds on continuous systems diagnosis, we present the different cases in turn to establish the connections between them, and the extensions required going to the more complex cases. We then describe a spectrum of online diagnoser implementations and illustrate how they trade off space and time complexity. We conclude with a summary and comparisons to work in

Table 6: Notation

Symbol	Explanation
f	A fault
F	The set of faults
F_i	A set of faults
m	A measurement
M	The set of measurements
M_i	A set of deviated measurements up to the ith event
q	A mode
Q	The set of modes
c	A candidate
C	The set of all candidates
d	A diagnosis
σ	An event
$\sigma_{f,m}$	A measurement deviation event
Σ_M	The set of measurement deviation events
σ_q	A controlled mode change event to mode q
Σ_Q	The set of controlled mode change events
λ	A trace
λ_i	A trace up to the ith event
$L_{f,M,q}$	The fault language for fault f, measurements M, and mode q
$\mathcal{L}_{f,M,q}$	The fault model for fault f, measurements M, and mode q
$\lambda_{c,q}$	A candidate trace for candidate c starting in mode q
$L_{c,M,q}$	The candidate language for candidate c, measurements M, starting in mode q
$\mathcal{D}^l_{F,M,Q}$	A diagnoser for faults F, measurements M, and modes Q, and candidate cardinality l

the DES area. The notation used throughout the previous chapters and this chapter is summarized in Table 6. The circuit example from Chapter VI (Fig. 39) will also be used throughout the chapter.

Diagnosability

Diagnosability is an important property of a system, because it enables us to make guarantees about the unique isolation of faults. We first provide definitions of *distinguishability* and *diagnosability* and then describe how these notions are captured in our event-based framework.

If two candidates will always produce different effects, we say they are distinguishable. For hybrid systems, we must define distinguishability with respect to an initial expected mode at the point of fault occurrence, i.e., that which will be used as the reference mode in computing deviations from expected behavior. In general, there may be a set of possible modes, which is why `RollBack` is necessary. To simplify the definitions, we define only with respect to a single expected mode of fault occurrence, but the framework can be extended to sets of possible expected modes.

Definition 37 (Distinguishability)**.** For an expected mode $q \in Q$ at the point of fault occurrence, a candidate c_i is distinguishable from a candidate c_j, denoted by $c_i \nsim_q c_j$, if for any possible sequence

of controlled mode changes, c_i always eventually produces effects on the measurements that c_j cannot.

In Chapters V and VI, we defined the notions of candidate languages for continuous and hybrid systems, respectively. These languages essentially capture the effects produced on the measurements for candidates, and thus characterize consistency of candidates with observed effects. Therefore, candidate languages can be used to establish distinguishability within our framework.

Recall that in single fault diagnosis for continuous systems, candidate languages are described by fault languages, $L_{f,M}$. If two faults share a common fault trace, or the fault trace of one fault is a prefix (denoted by $\sqsubseteq$) of a fault trace for the other fault, they are not distinguishable. For multiple faults in continuous systems, we form traces that interleave those of the single faults. In hybrid systems diagnosis, we also form interleaved traces of single faults, but these are also interleaved with controlled mode change events. Based on the language of a candidate, we can formally establish distinguishability in our framework.

Lemma 2. *For an expected mode $q_0 \in Q$ at the point of fault occurrence, a candidate c_i is* distinguishable *from a candidate c_j given measurements M and possible modes Q, if there does not exist a pair of candidate traces $\lambda_{c_i,q_0} \in L_{c_i,M,q_0}$ and $\lambda_{c_j,q_0} \in L_{c_j,M,q_0}$ such that $\lambda_{c_i} \sqsubseteq \lambda_{c_j}$.*

Proof. Assume c_i is not distinguishable from c_j, i.e., $c_i \sim_{q_0} c_j$ for initial mode of fault occurrence q_0. Then, by definition, starting in mode q_0, there must exist a maximal candidate trace by c_i that c_j can also produce. Therefore, there must exist some maximal candidate trace for c_i, i.e., some $\lambda_{c_i,q_0} \in L_{c_i,M,q_0}$, and some sequence of events for c_j that is not distinct from λ_{c_i,q_0}. So, λ_{c_i,q_0} must be a candidate trace λ_{c_j,q_0} for c_j. Therefore, if $c_i \sim_{q_0} c_j$ then there exits some $\lambda_{c_i,q_0} \in L_{c_i,M,q_0}$ and $\lambda_{c_j,q_0} \in L_{c_j,M,q_0}$ such that $\lambda_{c_i,q_0} \sqsubseteq \lambda_{c_j,q_0}$. By the contrapositive, if there does not exist $\lambda_{c_i,q_0} \in L_{c_i,M,q_0}$ and $\lambda_{c_j,q_0} \in L_{c_j,M,q_0}$ such that $\lambda_{c_i,q_0} \sqsubseteq \lambda_{c_j,q_0}$, then $c_i \nsim_{q_0} c_j$. □

Since candidate traces include mode change events, the candidate languages cover all possible sequences of controlled mode change events interleaved with measurement deviations. Therefore, checking distinguishability is equivalent to checking for common traces. So, if a maximal candidate trace, which is a sequence of controlled mode change events and measurement deviation events, for some candidate is a prefix for a second candidate, then if the first candidate occurs and produces that trace, the candidates cannot be distinguished, because no more measurements will deviate (since the trace is maximal).

As described in Chapter V, multiple faults occurring together can manifest in many different ways, and this reduces the distinguishability of the candidates. For example, a candidate (f_i, q_i)

can never be distinguished from another candidate $(f_i f_j, q_i)$, because f_i can completely mask f_j, or partially mask it in such a way that f_i by itself is also consistent, i.e., any trace for (f_i, q_i) is also a trace for $(f_i f_j, q_i)$. So, we cannot distinguish the candidates. As discussed in Chapter III, we are really only concerned with finding a minimal diagnosis, which leads us to the notion of *candidate coverage*.

Definition 38 (Candidate Coverage)**.** A candidate $c_i = (F_i, q)$ *covers* a candidate $c_j = (F_j, q)$ if $F_i \subseteq F_j$.

We say that a candidate *covers* another candidate if, for the same mode, its fault set is a nonstrict subset of the other candidate's fault set. So, for example, (f_1, q_0) covers $(f_1 f_2, q_0)$, but not $(f_1 f_2, q_1)$.

In our framework, a *system* can be defined as follows.

Definition 39 (System)**.** A *system* $\mathcal{S}$ is defined as $(F, M, Q, L_{F,M,Q})$, where $F = \{f_1, f_2, \ldots, f_n\}$ is a set of faults, $M = \{m_1, m_2, \ldots, m_p\}$ is a set of measurements, $Q = \{q_1, q_2, \ldots, q_r\}$ is a set of modes, and $L_{F,M,Q}$ is the set of fault languages for each fault in each mode, i.e., $L_{F,M,Q} = \{L_{f,M,q} : f \in F, q \in Q\}$.

Using candidate coverage, we obtain the following notion of l-fault diagnosability for a system.

Definition 40 (l-fault Diagnosability)**.** A system $\mathcal{S} = (F, M, Q, L_{F,M,Q})$ is *l-fault diagnosable* if for all c_i and c_j and possible modes of fault occurrence $q_0 \in Q$, where $|c_i| \leq l$ and $|c_j| \leq l$, if c_i does not cover c_j or c_j does not cover c_i then $c_i \nsim_{q_0} c_j$.

If the system is l-fault diagnosable, then for every pair of candidates with cardinality $\leq l$, where one does not cover the other, the candidates are distinguishable using the measurements in M. So, each sequence of measurement deviations and controlled mode changes we observe can be eventually linked to a minimal diagnosis with a unique candidate. Hence, we can uniquely isolate all candidates of interest. If the system is not diagnosable, then ambiguities may remain after fault isolation, i.e., after all possible measurement deviations have been observed.

The definition of l-fault diagnosability is about making guarantees about fault isolation. Although controlled mode change events affect the diagnosis, since the diagnoser has no control over which controlled mode change events are issued, we cannot, in general, make any restrictions about when a mode change event will be issued. So, l-fault diagnosability is conservative. It may be possible, however, to avoid ambiguous diagnosis results if certain mode changes are permitted or prohibited. We define this as Q-diagnosability.

Definition 41 (l-fault Q-diagnosability)**.** A system $\mathcal{S} = (F, M, Q, L_{F,M,Q})$ is *l-fault Q-diagnosable* if for all c_i and c_j and possible modes of fault occurrence $q_0 \in Q$, where $|c_i| \leq l$ and $|c_j| \leq l$, c_i does not cover c_j or c_j does not cover c_i, and $c_i \sim_{q_0} c_j$, then for every maximal λ_{c_i,q_0} where $\lambda_{c_i,q_0} \sqsubseteq \lambda_{c_j,q_0}$ for some λ_{c_j,q_0}, either there exists some sequence of controlled mode changes λ_Q where $\lambda_{c_i,q_0}\lambda_Q$ is not maximal for any candidate, or for every $\lambda_{c_k}\lambda_Q = \lambda_{c_i,q_0}$ where λ_Q is a sequence of controlled mode changes, λ_{c_k} is not maximal for any candidate.

Q-diagnosability means that for any trace that violates l-fault diagnosability, there is some sequence of controlled mode changes that can be applied such that the new trace is no longer maximal, i.e., more measurement deviations will occur, or for every partial trace that can become the violating trace via a sequence of controlled mode changes, the partial trace is not maximal. The first case corresponds to executing controlled mode changes to ensure more measurement deviations will occur. The second case corresponds to blocking a sequence of controlled mode changes such that we never encounter the violating trace in the first place.

Note that distinguishability and diagnosability are simplified for continuous systems, because controlled mode changes cannot take place. For SFD, distinguishability reduces to determining of the fault traces of single faults cannot be prefixes of another [86]. For MFD, we need to bring in candidate coverage because we are dealing with minimal diagnoses and only want to distinguish minimal diagnoses.

Other definitions of diagnosability have been provided in the literature (e.g., [6, 62, 76, 77, 134]), however, diagnosability is a property of a system using a modeling representation and a particular set of diagnosis algorithms. Therefore, our definitions are specific to our framework. However, there are similarities, e.g., in a DES framework, diagnosability is typically defined as there always being some finite sequence of observable events after a fault occurs that distinguishes the fault.

Diagnosers

We construct from our fault models an event-based diagnoser, which is an extended form of a finite automaton. If our system is diagnosable, we can construct a diagnoser that uniquely isolates all candidates. If not, the constructed diagnoser will give ambiguous results for some maximal traces. We wish to use the diagnoser to help determine system diagnosability as well as to serve as an efficient online implementation. The goal of the event-based diagnoser is, given a sequence of measurement deviation events and controlled mode change events, to determine which faults are consistent with the observed sequence. We define formally a *diagnoser* in our framework.

Definition 42 (Diagnoser)**.** A *diagnoser* for a fault set F, measurements M, and modes Q, with candidate cardinality limit l, is defined as $\mathcal{D}^l_{F,M,Q} = (S, I, \Sigma, \delta, A, D, Y)$ where S is a set of states, $I \subseteq S$ is set of initial states, Σ is a set of events, $\delta : S \times \Sigma \to S$ is a transition function, $A \subseteq S$ is a set of accepting states, $D \subseteq 2^C$ is a set of minimal diagnoses with candidates of cardinality $\leq l$, and $Y : S \to D$ is a diagnosis map.

A diagnoser is a finite automaton extended by a set of diagnoses and a diagnosis map. The initial states correspond to possible modes of fault occurrence, so for continuous systems, we replace I with a single initial state s_0. A diagnoser takes events as inputs, which, as with fault models, correspond to measurement deviations, but for hybrid systems, may also correspond to controlled mode change events. From the current state, a measurement deviation event causes a transition to a new state. The diagnosis for that new state represents the set of candidates that are consistent with the sequence of events seen up to the current point in time, i.e., it encodes the results that candidate tracking would obtain. So, like traditional DES diagnosers, the diagnoser states provide estimates of the system condition, but only after fault occurrence. As discussed in Chapter III, we assume that nominal behavior is tracked using a hybrid observer.

The accepting states of the diagnoser correspond to a fault isolation result. We say that a diagnoser *isolates* a candidate if it accepts all maximal candidate traces for the candidate, i.e., all traces in the candidate language, and the accepting states map to diagnoses containing the candidate or some other candidate that covers it (i.e., a candidate for the same mode which contains a subset of the candidate's faults).

Definition 43 (Isolation)**.** A diagnoser $\mathcal{D}^l_{F,M,Q}$ *isolates* a candidate c, where $|c| \leq l$, if for all possible initial modes of fault occurrence q_0, $\mathcal{D}^l_{F,M,Q}$ accepts all $\lambda_{c,q_0} \in L_{c,M,q_0}$ and for each $s \in A$ that accepts a $\lambda_{c,q_0} \in L_{c,M,q_0}$, there exists some $c' \in Y(s)$ such that c' covers c.

The notion of isolation gives us an indication of correctness of our diagnosers. If our diagnoser isolates all candidates, then it covers all possible observable candidate traces, and, therefore, is constructed correctly. If it does not isolate all candidates, then possible traces are omitted and incorrect diagnosis results will be obtained when those missing traces actually occur.

Due to fault masking, for a particular $\lambda_c \in L_{c,M,q_0}$, the candidate c may not be in the minimal diagnosis, which is why we define isolation with respect to minimal diagnoses. For a diagnoser to isolate a candidate, it is only required that for each trace, the diagnosis for that trace contains a candidate which covers c.

We also would like to achieve unique isolation of candidates, which is a stronger notion of isolation. Unique isolation relates to diagnosability, so provides guarantees about the ambiguity of the diagnosis results. For unique isolation, we seek unique candidates of cardinality $\leq l$ in the minimal diagnoses, which follows our definition of l-fault diagnosability.

Definition 44 (Unique Isolation)**.** A diagnoser $\mathcal{D}^l_{F,M,Q}$ *uniquely isolates* a candidate c, where $|c| \leq l$, if $\mathcal{D}^l_{F,M,Q}$ accepts all $\lambda_c \in L_{c,M,Q}$ and for each $s \in A$ that accepts some λ_c, $\{c'\} = Y(s)$ such that c' covers c.

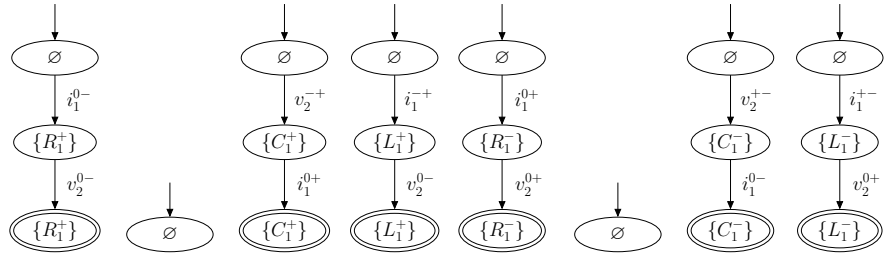
For unique isolation, we require the diagnoser isolates the candidate, but also that the corresponding accepting states uniquely determine c or a candidate that covers it. This means that the diagnoser will accept all maximal candidate traces, but also that each maximal trace will uniquely identify a minimal diagnosis with a single candidate. So, if we can design a diagnoser that isolates all candidates of interest, then by examining the diagnoser we can determine if it uniquely isolates all candidates, and if so, that the system is diagnosable. If not diagnosable, we can also use the diagnoser to determine which traces result in ambiguities, and if possible, be able to avoid those traces by blocking or executing certain controlled mode changes during isolation, i.e., determine if the system is l-fault Q-diagnosable.

Since candidate languages for hybrid systems must account for any possible sequence of controlled mode changes, the diagnoser must account for all these traces as well. In order to establish diagnosability of a hybrid system, this is unavoidable. The diagnoser, then, becomes infeasible for online diagnosis of practical hybrid systems. After describing how to construct the diagnoser for hybrid systems, we will describe alternative implementations that are more feasible.

Single Fault Diagnoser for Continuous Systems

In SFD for continuous systems, the system is defined as $\mathcal{S} = (F, M, L_{F,M})$, and we can simplify our representation of a candidate to a single fault rather than a set of faults, and omit the mode, i.e., $c = f$. Therefore, a state of the diagnoser maps to a set of consistent single faults, and only one initial state, s_0, is needed in the diagnoser. Since discrete faults only apply to hybrid systems, we also omit the discrete change feature from the signature since it provides no additional information for parametric faults alone.

Ultimately, we would like to systematically construct a diagnoser for a system $\mathcal{S}$ that isolates all $f \in F$. Further, we would like to show that if $\mathcal{S}$ is diagnosable, then this diagnoser *uniquely* isolates all $f \in F$. To do this, we first provide a way to construct a diagnoser for each fault f that isolates



(a) $\mathcal{D}^*_{\{R_1^+\}}$ (b) $\mathcal{D}^*_{\{R_2^+\}}$ (c) $\mathcal{D}^*_{\{C_1^+\}}$ (d) $\mathcal{D}^*_{\{L_1^+\}}$ (e) $\mathcal{D}^*_{\{R_1^-\}}$ (f) $\mathcal{D}^*_{\{R_2^-\}}$ (g) $\mathcal{D}^*_{\{C_1^-\}}$ (h) $\mathcal{D}^*_{\{L_1^-\}}$

Figure 46: Diagnosers for the individual faults of the circuit for mode q_0.

f. Then, we provide a composition operator to compose two diagnosers, such that if each diagnoser isolates its own set of faults, the composed diagnoser will isolate the combined set of faults. We then compose the individual diagnosers into a global diagnoser that isolates the complete set of system faults.

First, we construct a diagnoser for each single fault f from its fault model, $\mathcal{L}_{f,M}$. Because the fault model $\mathcal{L}_{f,M}$ accepts the fault language $L_{f,M}$, it is easy to show that this diagnoser isolates f.

Definition 45 ($\mathcal{D}^*_{\{f\},M}$)**.** Given f with $\mathcal{L}_{f,M} = (S, s_0, \Sigma, \delta, A)$, $\mathcal{D}^*_{\{f\},M} \triangleq (S, s_0, \Sigma, \delta, A, \{\{f\}\}, Y)$, where:

$$Y(s) = \begin{cases} \varnothing, & s = s_0 \\ \{f\}, & \text{otherwise.} \end{cases}$$

Lemma 3. $\mathcal{D}^*_{\{f\},M}$ *uniquely isolates* f*.*

Proof. $\mathcal{D}^*_{\{f\},M}$ extends $\mathcal{L}_{f,M}$ by defining D and Y. Therefore, by definition of $\mathcal{L}_{f,M}$, $\mathcal{D}^*_{\{f\},M}$ must accept all $\lambda_f \in L_{f,M}$. By definition of Y, $Y(s)$ for all $s \in A$ must map to $\{f\}$, since $s_0 \notin A$. So, $\mathcal{D}^*_{\{f\},M}$ uniquely isolates f. □

Example. The diagnosers corresponding to the individual faults of the circuit (Fig. 39) are shown in Figs. 46 and 47 for modes q_0 and q_1, respectively. The M subscript is omitted in the examples to simplify the notation. Note that since we assume that until a deviation is observed the system is nominal, initial states cannot be accepting states, so for R_2^+ and R_2^- in q_0, the initial state is not an accepting state. For example, consider $\mathcal{D}^*_{\{L_1^+\}}$ for q_0 and q_1. If the only mode is q_1, and L_1^+ occurs, then two possible traces can be produced, either $i_1^{-+} i_3^{0-} v_2^{0-}$ or $i_1^{-+} v_2^{0-} i_3^{0-}$. In q_0, however, i_3 cannot be affected, so only $i_1^{-+} v_2^{0-}$ is possible.

We next define a composition operator, denoted as Δ. The Δ composition provides a way to systematically construct the diagnoser for fault set F. We first define the *history* of a state.

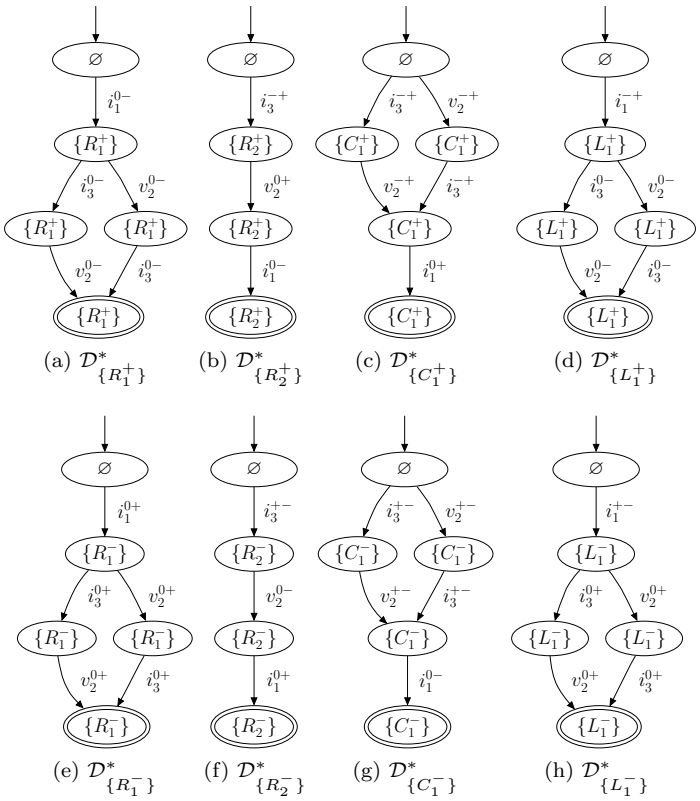


Figure 47: Diagnosers for the individual faults of the circuit for mode q_1.

Definition 46. The *history* of a state s is defined as $H(s_0) = \epsilon$ (the empty trace), and for $s \neq s_0$, $H(s) = \{\sigma_1\sigma_2 \ldots \sigma_n : \delta(\ldots(\delta(\delta(s_0, \sigma_1), \sigma_2)\ldots), \sigma_n) = s\}$.

Definition 47 (Δ Composition)**.** Given the diagnoser for a set of faults F_i, $\mathcal{D}_{F_i,M} = (S_i, s_{0,i}, \Sigma_i, \delta_i, A_i, D_i, Y_i)$ and the diagnoser for a set of faults F_j, $\mathcal{D}_{F_j,M} = (S_j, s_{0,j}, \Sigma_j, \delta_j, A_j, D_j, Y_j)$, the diagnoser defined by the Δ composition of $\mathcal{D}_{F_i,M}$ and $\mathcal{D}_{F_j,M}$ is $\mathcal{D}_{F_i,M}\Delta\mathcal{D}_{F_j,M} \triangleq (S, s_0, \Sigma, \delta, A, D, Y)$, where:

- $S = S_i \times S_j$,
- $s_0 = (s_{0,i}, s_{0,j})$,
- $\Sigma = \Sigma_i \cup \Sigma_j$,

- $\delta((s_i, s_j), \sigma) =$

$$\begin{cases} (\delta(s_i,\sigma), \delta(s_j,\sigma)), & \delta(s_i,\sigma) \wedge \delta(s_j,\sigma), \\ & |\lambda_i| = |\lambda_j|, \\ & \lambda_i \in H(s_i), \lambda_i \in H(s_j) \\ (\delta(s_i,\sigma), s_j), & \delta(s_i,\sigma) \wedge \neg\delta(s_j,\sigma) \\ (s_i, \delta(s_j,\sigma)), & \neg\delta(s_i,\sigma) \wedge \delta(s_j,\sigma) \\ \varnothing, & \text{otherwise,} \end{cases}$$

- $A = \{(s_i, s_j) : s_i \in A_i \vee s_j \in A_j\}$
- $D = \mathcal{P}(F_i \cup F_j)$
- $Y((s_i, s_j)) =$

$$\begin{cases} Y_i(s_i) \cup Y_j(s_j), & H(s_i) \cap H(s_j) \neq \varnothing \\ Y_i(s_i), & (\exists \lambda_i \in H(s_i), \lambda_j \in H(s_j)) \lambda_j \sqsubset \lambda_i \\ Y_j(s_j), & (\exists \lambda_i \in H(s_i), \lambda_j \in H(s_j)) \lambda_i \sqsubset \lambda_j \\ \varnothing, & \text{otherwise.} \end{cases}$$

Theorem 2. *If $\mathcal{D}_{F_i,M}$ isolates all $f_i \in F_i$, and $\mathcal{D}_{F_j,M}$ isolates all $f_j \in F_j$, then $\mathcal{D}_{F_i,M} \Delta \mathcal{D}_{F_j,M}$ isolates all faults in $F_i \cup F_j$.*

Proof. Assume $\mathcal{D}_{F_i,M}$ isolates all $f_i \in F_i$, and $\mathcal{D}_{F_j,M}$ isolates all $f_j \in F_j$. Then for some fault $f_i \in F_i$ with some trace $\lambda_{f_i} \in L_{f_i}$, $\mathcal{D}_{F_i,M}$ must accept λ_{f_i}, and this corresponds to some $s_{a,i} \in A_i$, where $f_i \in Y_i(s_{a,i})$. By the definition of δ, λ_{f_i} will correspond to state $(s_{a,i}, s_j)$ for some $s_j \in S_j$. Since $(s_{a,i}, s_j) \in A$ by definition of A, $\mathcal{D}_{F_i,M} \Delta \mathcal{D}_{F_j,M}$ accepts λ_{f_i}. By definition of H, $H(s_{a,i})$ must contain λ_{f_i}. If $\lambda_{f_i} \in H(s_j)$, then $H(s_{a,i}) \cap H(s_j)$ is nonempty, and $f_i \in Y_i(s_{a,i}) \subseteq Y((s_{a,i}, s_j))$. If $\lambda_{f_i} \notin H(s_j)$, then by definition of δ there must be some $\lambda_j \in H(s_j)$ where $\lambda_j \sqsubset \lambda_{f_i}$, so $f \in Y_i(s_{a,i}) = Y((s_{a,i}, s_j))$. Therefore, $\mathcal{D}_{F_i,M} \Delta \mathcal{D}_{F_j,M}$ isolates f_i. The same reasoning applies to some $f_j \in F_j$. Thus, $\mathcal{D}_{F_i,M} \Delta \mathcal{D}_{F_j,M}$ isolates all faults in $F_i \cup F_j$. □

The theorem shows that Δ preserves the isolation property. As a result, it is both commutative and associative with respect to isolation. Therefore, the order in which a set of diagnosers are composed does not matter, because at each intermediate step, isolation of the combined fault sets is maintained. Therefore, we can define the global diagnoser as a composition of the individual diagnosers.

Definition 48 ($\mathcal{D}^*_{F,M}$)**.** For fault set $F = \{f_1, f_2, \ldots, f_n\}$, $\mathcal{D}^*_{F,M}$ is defined as $\mathcal{D}^*_{\{f_1\},M} \Delta \mathcal{D}^*_{\{f_2\},M} \Delta \ldots \Delta \mathcal{D}^*_{\{f_n\},M}$.

Corollary 1. *The diagnoser $\mathcal{D}^*_{F,M}$ isolates all $f \in F$.*

Because each $\mathcal{D}^*_{\{f\},M}$ isolates f if constructed from $\mathcal{L}_{f,M}$ as described, and since Δ preserves the isolation property, then $\mathcal{D}^*_{F,M}$ as constructed above isolates all $f \in F$. Further, if the fault set is diagnosable, then this diagnoser guarantees that each fault is uniquely isolated.

Theorem 3. *A system $\mathcal{S} = (F, M, L_{F,M})$ is diagnosable if and only if $\mathcal{D}^*_{F,M}$ uniquely isolates all $f \in F$.*

Proof. Assume some $f_i \in F$ with fault trace $\lambda_{f_i} \in L_{f_i,M}$. $\mathcal{D}^*_{F,M}$ accepts λ_{f_i} and for corresponding accepting state s_a, $f_i \in Y(s_a)$ by Corollary 1 and the definition of isolation. Since $\mathcal{S}$ is diagnosable, there is no $f_j \in F$ with fault trace $\lambda_{f_j} \in L_{f_j,M}$ where $\lambda_{f_i} \sqsubseteq \lambda_{f_j}$. Therefore, $f_j \notin Y(s_a)$. So, $Y(s_a) = \{f_i\}$ and $\mathcal{D}^*_{F,M}$ uniquely isolates each $f \in F$. Assume that $\mathcal{D}^*_{F,M}$ uniquely isolates each $f \in F$. Then each possible fault trace λ_{f_i} has an associated accepting state s_a, where $Y(s_a) = \{f_i\}$. Thus, there cannot be some $\lambda \sqsubseteq \lambda_{f_j}$ for $f_i \neq f_j$ that can reach s_a, otherwise $f_j \in Y(s_a)$. Therefore, $f_i \nsim f_j$, so F is diagnosable. Thus S is diagnosable if and only if $\mathcal{D}^*_{F,M}$ uniquely isolates each $f \in F$. □

The diagnoser construction procedure is shown as Algorithm 12. It is described as combining two diagnosers, but, as with Δ, can be easily be modified to combine k diagnosers simultaneously. The construction algorithm operates by tracing paths in the two given diagnosers. If the same event is available in both current states, then we advance in both diagnosers, i.e., $(s_1, s_2) \xrightarrow{\sigma} (\delta(s_1, \sigma), \delta(s_2, \sigma))$. Otherwise, we advance in only one, e.g., if σ can only be taken from s_1, then $(s_1, s_2) \xrightarrow{\sigma} (\delta(s_1, \sigma), s_2)$. If the computed diagnosis for the new state, d, is empty, then $\delta((s_1, s_2), \sigma)$ is set to $\varnothing$, because this means the current sequence of measurement deviations is inconsistent with the single fault assumption.

The diagnosis for the new state is formed by composing the current diagnosis with the hypothesis set. The hypothesis set, h, is the set of faults consistent with the current event. It is formed as the union of the diagnoses of the diagnoser states advanced to via the event σ. The new diagnosis for the composed diagnoser state is constructed as the intersection of the current diagnosis and the hypothesis set. For example, if $\{f_i, f_j\}$ is the current diagnosis and the hypothesis set is $\{f_i\}$ then the new diagnosis is $\{f_i\}$, which means that only f_i is consistent with the current event sequence.

Example. The diagnosers for the circuit (Fig. 39) are shown in Figs. 48 and 49 for modes q_0 and q_1, respectively. We can see that since all accepting states have singleton diagnoses, the system is diagnosable.

Algorithm 12 $\mathcal{D} \leftarrow \mathcal{D}_1 \Delta \mathcal{D}_2$

$S \leftarrow \varnothing$, $\delta \leftarrow \varnothing$, $D \leftarrow \varnothing$, $\Sigma \leftarrow \Sigma_1 \cup \Sigma_2$
$s_o \leftarrow (s_{o1}, s_{o2})$, $Y(s_o) \leftarrow \varnothing$, $S \leftarrow \{s_o\}$, $S_{pend} \leftarrow \{s_o\}$, $A \leftarrow \varnothing$
while $S_{pend} \neq \varnothing$ **do**
  $(s_1, s_2) \leftarrow \texttt{pop}(S_{pend})$
  for all $\sigma \in \Sigma$ **do**
    if $\delta_1(s_1, \sigma)$ **and** $\delta_2(q_2, \sigma)$ **then**
      $s' \leftarrow (\delta_1(s_1, \sigma), \delta_2(s_2, \sigma))$
      $h \leftarrow Y(\delta_1(s_1, \sigma)) \cup Y(\delta_2(s_2, \sigma))$
    else if $\delta_1(s_1, \sigma)$ **then**
      $s' \leftarrow (\delta_1(s_1, \sigma), s_2)$
      $h \leftarrow Y(\delta_1(s_1, \sigma))$
    else if $\delta_2(q_2, \sigma)$ **then**
      $s' \leftarrow (s_1, \delta_2(s_2, \sigma))$
      $h \leftarrow Y(\delta_2(s_2, \sigma))$
    else
      $s' \leftarrow \varnothing$
      $h \leftarrow \varnothing$
    if $s' \neq \varnothing$ **then**
      if $Y((s_1, s_2)) = \varnothing$ **then**
        $d \leftarrow h$
      else
        $d \leftarrow Y((s_1, s_2)) \cap h$
      if $d \neq \varnothing$ **then**
        $S \leftarrow S \cup \{s'\}$
        $\delta((s_1, s_2), \sigma) \leftarrow s'$
        $D \leftarrow D \cup \{d\}$
        $Y(s') \leftarrow d$
        if $s_1 \in A_1$ **or** $s_2 \in A_2$ **then**
          $A \leftarrow A \cup \{s'\}$
        if $s' \notin S_{pend}$ **then**
          $\texttt{push}(S_{pend}, s')$

Consider the fault trace $i_3^{-+} v_2^{0+} i_1^{0-}$ for mode q_1 (Fig. 49). For i_3^{-+} occurring as the first deviation, only C_1^+ or R_2^+ could have occurred, given the known fault signatures and measurement orderings. Therefore, the new diagnosis is $\{C_1^+, R_2^+\}$. For v_2^{0+} occurring next, of our current faults, only R_2^+ is consistent, therefore, our new diagnosis is the intersection of $\{C_1^+, R_2^+\}$ and $\{R_2^+\}$, which is $\{R_2^+\}$. At this point we obtain a unique fault hypothesis. The only possible measurement deviation remaining is i_1^{0-} which must still be consistent with $\{R_2^+\}$.

Multiple Fault Diagnoser for Continuous Systems

In MFD for continuous systems, the system is defined as $\mathcal{S} = (F, M, L_{F,M})$, and we can simplify our representation of a candidate by omitting the mode, i.e., $c = F_i$ where $F_i \subseteq F$. Therefore, a state of the diagnoser maps to a set of consistent multiple fault hypotheses, and only one initial state, s_0, is needed in the diagnoser. Since discrete faults only apply to hybrid systems, we also omit the

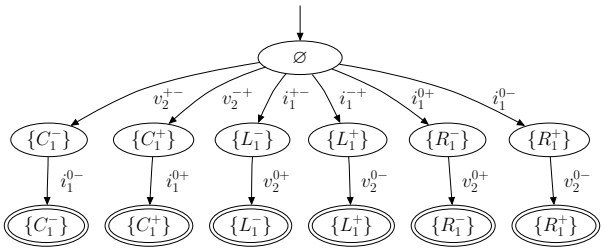


Figure 48: Event-based diagnoser for the circuit for mode q_0.

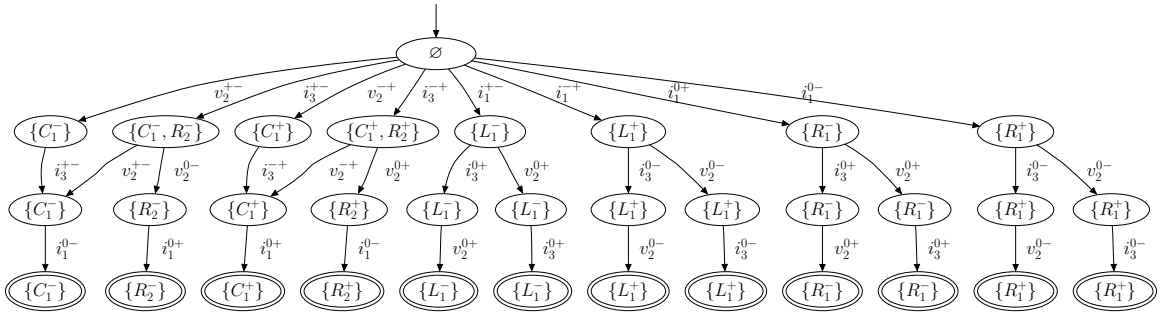


Figure 49: Event-based diagnoser for the circuit for mode q_1.

discrete change feature from the signature since it provides no additional information for parametric faults alone.

Ultimately, we would like to systematically construct a diagnoser for a system $\mathcal{S}$ that isolates all possible candidates of cardinality $\leq l$. Further, we would like to show that if $\mathcal{S}$ is diagnosable, then this diagnoser *uniquely* isolates all such candidates. To do this, we use the $\mathcal{D}^*_{f,M}$ constructed in the previous section, and provide a composition operator to *simultaneously* compose all the diagnosers to a global diagnoser that isolates all the valid candidates.

As described in Chapter V, when multiple faults occur together they produce additional traces that are formed by the interleaving of single fault traces due to masking effects and different times of individual fault occurrence. In incremental consistency checking, we project out measurements that have already deviated to obtain the set of consistent candidates for a new observation, i.e., the hypothesis set. For a diagnoser, the state-based form of the measurement projection operation on traces is formalized using *boundaries* and *boundary transition functions*.

Definition 49 (Boundary)**.** The *boundary* for a state s and deviated measurements M_i, $B_{M_i}(s)$, is defined as the set of all states $\delta(\lambda, s)$ such that $P_{M-M_i}(\lambda) = \epsilon$.

The boundary for a state s is basically the set of states that may be transitioned to from s via a trace λ consisting of only events for measurements that have already deviated, i.e., measurements

corresponding to the events for traces in the history of the state. Using the notion of a boundary, we define a *boundary transition function* with respect to a set of deviated measurements.

Definition 50 (Boundary Transition Function)**.** The *boundary transition function* for an event σ, state s, and set of deviated measurements M_i, $\delta_{M_i}(\sigma, s)$, is a transition function that maps σ and s to some state s', such that $s' = \varnothing$ if the cardinality of $\{\delta(\sigma, s_B) : s_B \in B_{M_i}(s)\}$ is not 1, or s' is the single element in $\{\delta(\sigma, s_B) : s_B \in B_{M_i}(s)\}$, otherwise.

In other words, $\delta_{M_i}(\sigma, s)$ returns the unique state that can be reached from a boundary state of s via σ, or $\varnothing$ if there are no states that can be reached or if the state is not unique. Because of the way the $\mathcal{D}^*_{\{f\},M}$ diagnosers are computed, the reachable state will always be unique or null, because traces with the same set of measurements map to the same state. So, the different histories of a state in these diagnosers have the same set of measurements. In the following, we denote the measurements that have deviated in a state s as $M(s)$.

Due to the need for boundaries and boundary transition functions, we define our Δ_l composition as composing all n diagnosers *simultaneously*, rather than incrementally as in the SFD case. By selecting $l = 1$, Δ_1 becomes the Δ composition extended to combine all diagnosers simultaneously.

Definition 51 (Δ_l Composition)**.** Given the n individual $\mathcal{D}^*_{\{f_i\},M} = (S_i, s_{0,i}, \Sigma_i, \delta_i, A_i, D_i, Y_i)$ for each $f \in F$, $\mathcal{D}^{l*}_{F,M} \triangleq \Delta_l(\mathcal{D}^*_{\{f_1\},M}, \mathcal{D}^*_{\{f_2\},M}, \ldots, \mathcal{D}^*_{\{f_n\},M})$, where

- $s_0 = (s_{0,1}, s_{0,2}, \ldots, s_{0,n}, \varnothing)$
- $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_n$
- $\delta(\sigma, (s_{i,1}, s_{i,2}, \ldots, s_{i,n}, d_i)) = (s_{i+1,1}, s_{i+1,2}, \ldots, s_{i+1,n}, d_{i+1})$, where
  - $M_i \triangleq M((s_{i,1}, s_{i,2}, \ldots, s_{i,n}, d_i))$,
  - $\forall j, s_{i+1,j} = s_{i,j}$ if $\delta_{M_i,j}(\sigma, s_{i,j}) = \varnothing$, or $\delta_{M_i,j}(\sigma, s_{i,j})$ otherwise, and
  - $d_{i+1} = d_i \wedge_l h_{F,M_i}(\sigma) \neq \varnothing$
- S is the set of all s reachable through δ from s_0
- D is the set of all d_i in each $(s_{i,1}, s_{i,2}, \ldots, s_{i,n}, d_i) \in S$
- A is the set of all $s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n}, d_i) \in S$ where there exists some $s_{i,j} \in s_i$, with some $s_{B,j} \in B_{M(s_i)}(s_{i,j})$[1] where $s_{B,j} \in A_j$, such that the $c \in Y_j(s_{B,j})$ covers some $c' \in Y(s_i)$

[1]Recall that $M(s)$ denotes the set of measurements that have deviated up to state s.

- $Y((s_{i,1}, s_{i,2}, \ldots, s_{i,n}, d_i)) = d_i$

Note that in the Δ_l composition, states are also indexed by the associated diagnosis. Because we project out measurements which have already deviated, we may reach the same states in the individual diagnosers, but obtain different diagnoses depending on which specific deviations were observed for the measurements. Therefore a state is uniquely identified by both the corresponding states of the individual diagnosers and the diagnosis.

We now proceed to show that $\mathcal{D}^{l*}_{F,M}$ isolates all valid candidates, i.e., it accepts all maximal candidate traces, and the diagnoses of the accepting states contain a candidate that covers the valid candidate. Recall that for multiple faults in continuous systems, the language of a candidate consists of all traces where each successive event of the trace is consistent with one of the faults in the candidate, for the measurements that have not yet deviated.

Theorem 4. *The diagnoser $\mathcal{D}^{l*}_{F,M}$ isolates all candidates of cardinality $\leq l$.*

Proof. Assume candidate c, with $|c| \leq l$, and trace $\lambda = \sigma_1\sigma_2 \ldots \sigma_k \in L_{c,M}$. By definition of $L_{c,M}$, λ must be consistent with c, so, therefore, $h_{F,M_i}(\sigma_i)$ will always contain some $f \in c$ for any $\sigma_i \in \lambda$. Thus, every time $\wedge_l$ is used, the resultant diagnosis will contain some c' that covers c since $|c| \leq l$, by definition of $\wedge_l$, and, therefore d_i will never be empty. So, by definition of δ, there will be a corresponding state s for λ. So, some c' will be in $Y(s)$, where c' covers c. State s corresponds to some $(s_1, s_2, \ldots, s_n, d)$. Since $Y(s)$ contains c' which covers c there must be some $f \in c'$ where $f \in c$. So there must be a substate in s that corresponds to $\mathcal{D}^*_{f,M}$ where for the boundary of that state, there are no measurement deviations possible, therefore, it must be an accepting state in $\mathcal{D}^*_{f,M}$. By definition of A, then, $s \in A$. Since λ was general, $\mathcal{D}^{l*}_{F,M}$ isolates c. Since c was general, $\mathcal{D}^{l*}_{F,M}$ isolates all candidates of cardinality $\leq l$. □

Further, we can show that if the system $\mathcal{S}$ is l-fault diagnosable, then the diagnoser uniquely isolates all candidates of cardinality $\leq l$.

Theorem 5. *A system $\mathcal{S} = (F, M, L_{F,M})$ is l-fault diagnosable if and only if $\mathcal{D}^{l*}_{F,M}$ uniquely isolates all c, where $|c| \leq l$.*

Proof. Assume $\mathcal{S}$ is l-fault diagnosable. Assume a c and $\lambda \in L_{c,M}$, with $|c| \leq l$. $\mathcal{D}^{l*}_{F,M}$ isolates c, so must have corresponding accepting state s with $c' \in Y(s)$ where c' covers c. Since $\mathcal{S}$ is diagnosable, there cannot be a c'' that does not cover c or that c does not cover where c and c'' are not distinguishable, by definition of l-fault diagnosability. So, there cannot be some common prefix λ that maps to an accepting state that has both a c' that covers c and a c''' that covers c''. So, $\mathcal{D}^{l*}_{F,M}$

uniquely isolates all c, where $|c| \leq l$. Assume $\mathcal{D}^{l*}_{F,M}$ uniquely isolates all c, where $|c| \leq l$. Then each possible fault trace λ has an accepting state s where $c' \in Y(s)$ and c' covers c. Thus, there cannot be some c'' that does not cover c or that c does not cover, with trace λ'' that reaches the same state, otherwise c'' or some candidate that covers it is in $Y(s)$. Therefore, c and c'' are distinguishable, so $\mathcal{S}$ is l-fault diagnosable. Thus $\mathcal{S}$ is l-fault diagnosable if and only if $\mathcal{D}^{l*}_{F,M}$ uniquely isolates all c, where $|c| \leq l$. □

An implementation of Δ_l is given as Algorithm 13. It tracks the given diagnosers and incrementally builds up the composed diagnoser. Starting from a particular state and for a particular event, the algorithm advances in each given diagnoser from the boundary of each current diagnoser state. For the diagnosers that are advanced in, their faults are added to the hypothesis set. The new diagnosis is then created using $\wedge_l$, and if the diagnosis is valid, the new state, new transition, and new diagnosis are updated. To properly index the state, the diagnosis of the state is required, not just the individual states of the composing diagnosers. Therefore, the new diagnosis is set as a component to the new state. The state is made an accepting state if there is a current boundary state in one of the composing diagnosers that is accepting and its candidate is covered by the composed state's diagnosis.

Example. Consider $F = \{C_1^+, R_2^+\}$ for mode q_1. The corresponding diagnoser is shown in Fig. 50a, for $l = 2$. It contains the three valid traces for the single faults, which all map to accepting states identifying the single faults as minimal diagnoses. For example, consider the fault trace $i_3^{-+} v_2^{-+} i_1^{0+}$. After i_3^{-+} occurs, the diagnosis is $\{C_1^+, R_2^+\}$, i.e., either C_1^+ occurred or R_2^+ occurred, and they may have occurred together but the diagnosis $\{C_1^+, R_2^+, C_1^+ R_2^+\}$ is not minimal. After v_2^{-+}, R_2^+ by itself is no longer consistent, so the hypothesis set is $\{C_1^+\}$ and the diagnosis is $\{C_1^+, C_1^+ R_2^+\}$, but we remove $C_1^+ R_2^+$ to make the diagnosis minimal. After i_1^{0+}, C_1^+ is still consistent by itself so remains the diagnosis. If i_1^{0-} had occurred instead, the hypothesis set would be $\{R_2^+\}$, so the resultant diagnosis would be $\{C_1^+ R_2^+\}$, i.e., it must be that both faults occurred together, and R_2^+ masked the effect of C_1^+ on i_1.

Consider $F = \{L_1^+, R_1^+\}$ for mode q_1. The corresponding diagnoser is shown in Fig. 50b, for $l = 2$. Since the only difference between the two single faults is the deviation of the first measurement, i_1, the diagnoser splits at the first state then has the same behavior, but with different diagnoses. From this diagnoser, since the minimal diagnoses are all single faults, then this implies that $L_{L_1^+ R_1^+, M} = L_{L_1^+, M} \cup L_{R_1^+, M}$, i.e., we can never determine if both faults had to have occurred together. Based

Algorithm 13 $\mathcal{D}^{l*}_{F,M} \leftarrow \Delta_l(\mathcal{D}^*_{\{f_1\},M}, \mathcal{D}^*_{\{f_2\},M}, \ldots, \mathcal{D}^*_{\{f_n\},M})$

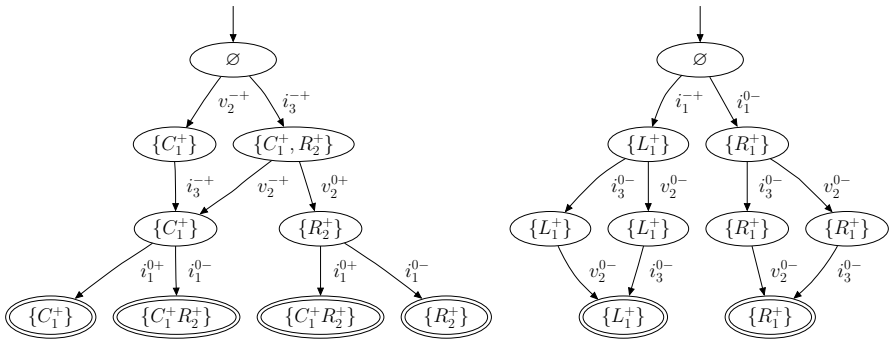
$S \leftarrow \varnothing$
$\delta \leftarrow \varnothing$
$D \leftarrow \varnothing$
$\Sigma \leftarrow \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_n$
$s_0 \leftarrow (s_{0,1}, s_{0,2}, \ldots, s_{0,n}, \varnothing)$
$Y(s_0) \leftarrow \varnothing$
$S_{pend} \leftarrow \{s_0\}$
while $S_{pend} \neq \varnothing$ **do**
  $s \leftarrow \texttt{pop}(S_{pend})$
  for all $\sigma \in \Sigma$ **do**
    $h \leftarrow \varnothing$
    for all $s_i \in s$ **do**
      if $\delta_{M(s)}(\sigma, s_i) \neq \varnothing$ **then**
        $s'_i \leftarrow \delta_{M(s)}(\sigma, s_i)$
        $h \leftarrow h \cup Y_i(s_i)$
      else
        $s'_i \leftarrow s_i$
    $d \leftarrow Y(s) \wedge_l h$
    if $d \neq \varnothing$ **then**
      $d' \in s' \leftarrow d$
      $\texttt{push}(S_{pend}, s')$
      $S \leftarrow S \cup \{s'\}$
      $\delta(s, \sigma) \leftarrow s'$
      for all $s'_i \in s'$ **do**
        if $\exists s_B \in B_{M(s')}(s'_i), s_B \in A_i$ **and** $c \in Y_i(s_B)$ covers some $c' \in Y(s')$ **then**
          $A \leftarrow A \cup \{s'\}$
          break
      $D \leftarrow D \cup \{d\}$
      $Y(s') \leftarrow d$

on our definition of l-fault diagnosability, however, we still consider the system 2-fault diagnosable for $F = \{L_1^+, R_1^+\}$, because we always obtain a unique minimal diagnosis.

The following example illustrates some diagnosability results for the circuit.

Example. The diagnoser for $F = \{C_1^+, L_1^-, R_1^-\}$ with $l = 2$ for mode q_0 is shown in Fig. 51. The diagnoser shows that the system for the given F is 2-fault diagnosable. In each accepting state there is a unique minimal diagnosis. Only two fault traces may arise in which we can be sure that two faults have occurred together. We can also say the system is 3-fault diagnosable, however since only two measurements are involved, we will never obtain a minimal diagnosis with a candidate consisting of three faults.

The diagnoser for $F = \{C_1^+, L_1^-, R_2^+\}$ with $l = 2$ for mode q_1 is shown in Fig. 52. Consider the trace $i_3^{-+} i_1^{+-} v_2^{-+}$. After i_3^{-+}, the initial diagnosis is $\{C_1^+, R_2^+\}$. After i_1^{+-}, the hypothesis set is $\{L_1^-\}$, so now there will be two consistent double faults, i.e., the diagnosis becomes $\{C_1^+ L_1^-, L_1^- R_2^+\}$. After v_2^{-+}, the hypothesis set is $\{C_1^+\}$, so the diagnosis becomes $\{C_1^+ L_1^-, C_1^+ L_1^- R_2^+\}$, which is pruned to $\{C_1^+ L_1^-\}$, because the candidate $\{C_1^+, L_1^-, R_2^+\}$ violates $l = 2$ and is covered by another candidate



(a) $\mathcal{D}^{2*}_{C_1^+ R_2^+}$ (b) $\mathcal{D}^{2*}_{L_1^+ R_1^+}$

Figure 50: Selected diagnosers for mode q_1 with $l = 2$.

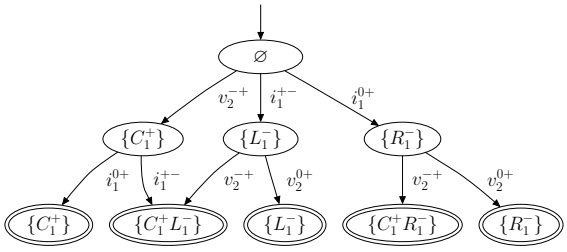


Figure 51: Diagnoser for $F = \{C_1^+, L_1^-, R_1^-\}$ in mode q_0 with $l = 2$.

in the diagnosis. If v_2^{0+} had occurred instead, then the hypothesis set would be $\{L_1^-, R_2^+\}$, and the diagnosis would not change. Therefore, the diagnoser shows us that the system is not 2-fault diagnosable. In fact, there are two different accepting states where the diagnosis result is ambiguous, i.e., it is unknown which double fault has occurred. However, the diagnoser does illustrate the traces that lead to these states, and, further, in both of these states, it is at least certain that L_1^- is one of the faults that occurred.

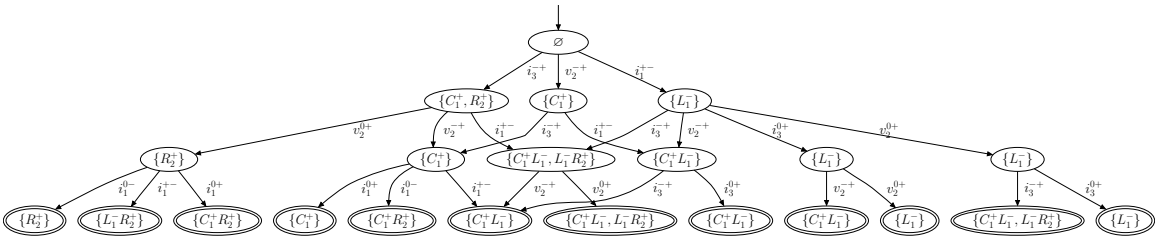


Figure 52: Diagnoser for $F = \{C_1^+, L_1^-, R_2^+\}$ in mode q_1 with $l = 2$.

Diagnoser for Hybrid Systems

We now design diagnosers for hybrid systems, where the system is defined as $\mathcal{S} = (F, M, Q, L_{F,M,Q})$, and we use the full representation of a candidate, i.e., $c = (F_i, q_i)$. In hybrid systems, both controlled and autonomous mode changes (including discrete faults) may occur. As discussed in Chapter VI, we analyze nominal autonomous mode changes after fault occurrence the same way we analyze discrete faults, i.e., we predict their effects and isolate them along with faults. Therefore, F includes parametric faults, discrete faults, and unpredicted nominal autonomous mode changes. Again, this is a different approach than Hybrid TRANSCEND, because we maintain only a single reference of nominal behavior that updates only with controlled mode changes, and assume mode changes do not cause state resets. It also assumed that we can predict the measurement deviations that will be produced by autonomous mode changes as we do with discrete faults. We leave diagnoser design for the more general approach as future work.

Ultimately, we would like to systematically construct a diagnoser for a hybrid system $\mathcal{S}$ that isolates all possible candidates. Further, we would like to show that if $\mathcal{S}$ is diagnosable, then this diagnoser *uniquely* isolates all candidates. To do this, we use individual diagnosers like the $\mathcal{D}^*_{\{f\},M}$ constructed previously for continuous systems, but which include the modes in the diagnoses of the states. We then provide a composition operator to simultaneously compose all the diagnosers to a global diagnoser that isolates all the valid candidates.

First, we construct a diagnoser, $\mathcal{D}^*_{\{f\},M,q}$ for each single fault f within each mode q from $\mathcal{L}_{f,M,q}$.

Definition 52 ($\mathcal{D}^*_{\{f\},M,q}$)**.** Given fault f and mode q for measurements M, with $\mathcal{L}_{f,M,q} = (S, s_0, \Sigma, \delta, A)$, $\mathcal{D}^*_{\{f\},M}$ is defined as $(S, s_0, \Sigma, \delta, A', \{\{(f,q)\}\}, Y)$, where $Y(s) = \{(f,q)\}$ for all $s \in S$, and $A' = A$ if $S \neq \{s_0\}$, or $A' = \{s_0\}$ otherwise.

These diagnosers are different from those for continuous systems in two ways. First, the diagnosis for the states contains the mode. Second, the initial state also is mapped to the fault, and the initial state may also be an accepting state. The second change is necessary because if our diagnosis contains a fault, and we change to a mode where the fault has no future effects, this corresponds to maximal candidate trace for the new mode, and, therefore, must correspond to an accepting state in the diagnoser.

Example. The extended diagnosers corresponding to the individual faults in the different modes of the circuit are shown in Fig. 53. Since we must enumerate all possible traces, we need diagnosers for the faults in both the nominal and the fault modes, e.g., we require both $\mathcal{D}^*_{R_1^+,q_0}$ and $\mathcal{D}^*_{R_1^+,q_{\tau_0}}$, but the diagnosers for the parametric faults in the discrete fault are not shown since the diagnosers for

the faulty modes are the same as for the nominal mode, only with the mode changed. For example, $\mathcal{D}^*_{R_1^+,q_{\tau_0}}$ looks just like $\mathcal{D}^*_{R_1^+,q_0}$, only with q_0 replaced with q_{τ_0} in the diagnoses.

Consider the diagnosers $\mathcal{D}^*_{L_1^+,q_0}$ and $\mathcal{D}^*_{L_1^+,q_1}$. As in the continuous systems case, if the mode is q_1, and L_1^+ occurs, then two possible traces can be produced, either $i_1^{-+,X} i_3^{0-,X} v_2^{0-,X}$ or, alternatively, $i_1^{-+,X} v_2^{0-,X} i_3^{0-,X}$. In q_0, i_3 cannot be affected, so only $i_1^{-+,X} v_2^{0-,X}$ is possible. But, if a controlled mode change occurs, it may change the observed effects. For example, if we start in q_1 and see $i_1^{-+,X} v_2^{0-,X}$, but then the mode is changed to q_0, we will not observe the effect on i_3 because the causal link between the fault and the measurement does not exist in the new mode. Therefore, we will observe no new measurement deviations. If, however, we were in q_1, observed $i_1^{-+,X} i_3^{0-,X}$, and changed to q_0, we would see $v_2^{0-,X}$ because in mode q_0, L_1^+ can still affect v_2.

Some differences emerge in the diagnosers for hybrid systems as compared to those for continuous systems. As described in Chapter VI, in hybrid systems we have to explicitly deal with both measurement deviation events (in Σ_M) and controlled mode change events (in Σ_Q). Since we want to use the diagnoser to analyze system diagnosability, and since this relates to candidate languages, mode change events must be included in the diagnoser, because they affect the diagnosis. Recall that given modes $Q = \{q_0, q_1, \ldots, q_r\}$, we abstract the mode change events to $\Sigma_Q = \{\sigma_{q_0}, \sigma_{q_1}, \ldots, \sigma_{q_r}\}$, where σ_{q_i} indicates an event that changes the system to mode q_i. The sequence of the commands should be consistent with the discrete mode behavior of the system, which can be described by the mode transition function μ. For example, σ_{q_1} would probably not be immediately followed by σ_{q_1} because the controller has memory, i.e., $\mu(\sigma_{q_1}, q_1) = \varnothing$.

A second important difference is that there may be different possible modes of fault occurrence, depending on the history of control actions. Consistency of candidates depends on the expected mode. For example, in the circuit, if the mode is known to be q_1 immediately before fault occurrence, then the stuck-on fault, τ_1, should not be considered as a fault candidate, because if it did occur we would not have seen a deviation, because q_1 and q_{τ_1} map to the same mode of the corresponding junction, i.e., the junction is on for both modes. We must create an initial state in the diagnoser for each possible mode of fault occurrence.

We now describe a composition operator, Π_l, that simultaneously combines the $\mathcal{D}^*_{\{f\},M,q}$ for each possible (f, q) pair. To simplify, we describe the composition assuming that the mode immediately before fault occurrence, $q(t_f^-)$, is always known, i.e., `RollBack` is not needed to generate $q(t_f^-)$. We split the mode set Q into nominal modes Q_N and fault modes Q_F.

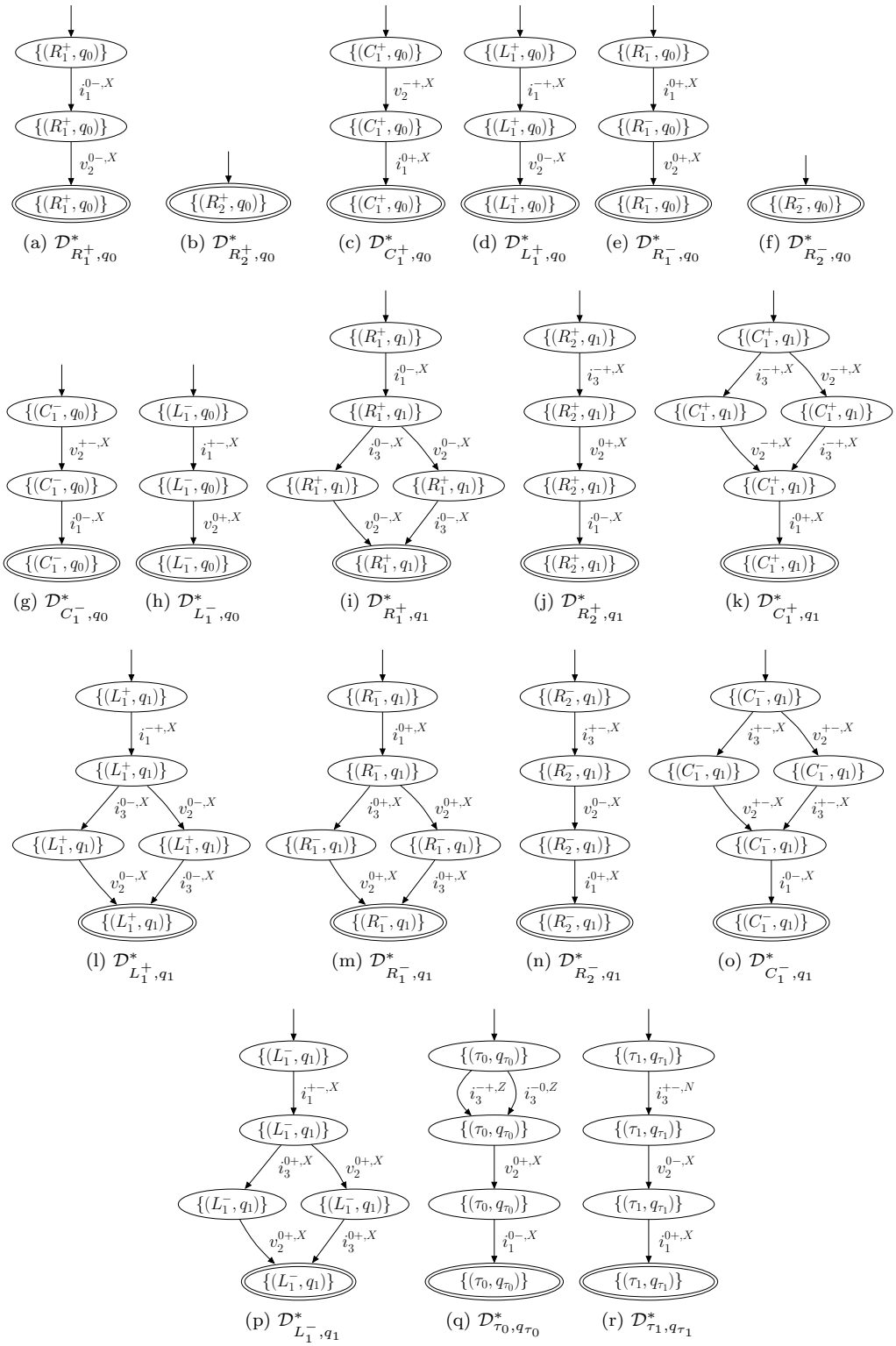


Figure 53: Hybrid diagnosers for the individual faults of the circuit.

Definition 53 (Π_l Composition)**.** Given the set of all k (f, q) diagnosers, $\mathbb{D} = \{\mathcal{D}^*_{\{f\},M,q} : f \in F, q \in Q\}$, $\mathcal{D}^{l*}_{F,M,Q} \triangleq \Pi_l(\mathbb{D})$, where

- $I = \{(s_{0,1}, s_{0,2}, \ldots, s_{0,k}, q, (\varnothing, q)) : q \in Q_N\}$
- $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_k \cup \Sigma_Q$
- $\delta(\sigma, (s_{i,1}, s_{i,2}, \ldots, s_{i,k}, q_i, d_i)) = (s_{i+1,1}, s_{i+1,2}, \ldots, s_{i+1,k}, q_{i+1}, d_{i+1})$, where
  - $\sigma \in \Sigma_Q$
  - all $s_{i+1,j} = s_{i+1,j}$
  - $q_{i+1} = \mu(\sigma, q_i)$
  - $d_{i+1} = \{(f, \mu(\sigma, q)) : \mu(\sigma, q) \neq \varnothing \wedge (f, q) \in d_i\}$
- $\delta(\sigma, (s_{i,1}, s_{i,2}, \ldots, s_{i,k}, q_i, d_i)) = (s_{i+1,1}, s_{i+1,2}, \ldots, s_{i+1,k}, q_{i+1}, d_{i+1})$, where
  - $\sigma \in \Sigma_M$
  - $q_{i+1} = q_i$
  - $M_i = M((s_{i,1}, s_{i,2}, \ldots, s_{i,k}, q_i, d_i))$,
  - $s_{i+1,j} = s_{i,j}$ if $\delta_{M_i,j}(\sigma, s_{i,j}) = \varnothing$, or $\delta_{M_i,j}(\sigma, s_{i,j})$ otherwise, and
  - $d_{i+1} = d_i \wedge_l h_{F,M_i}(\sigma) \neq \varnothing$
- S is the set of all s reachable through δ from some $s_0 \in I$
- A is the set of all $s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,k}, q_i, d_i) \in S$ where there exists some $s_{i,j} \in s_i$, with some $s_{B,j} \in B_{M(s_i)}(s_{i,j})$ where $s_{B,j} \in A_j$, such that the $c \in Y_j(s_{B,j})$ covers some $c' \in Y(s_i)$
- D is the set of all d_i in each $(s_{i,1}, s_{i,2}, \ldots, s_{i,k}, q_i, d_i) \in S$
- $Y((s_{i,1}, s_{i,2}, \ldots, s_{i,k}, q_i, d_i)) = d_i$

In the Π_l composition, states are not only indexed by the states of the individual diagnosers and the current diagnosis, but also by the expected mode. This corresponds to the mode that the controller thinks the system is in. It is independent of the candidates in the current diagnosis, because it is not changed when an event $\sigma \in \Sigma_M$ is received. Indexing by this mode is important because it determines which new controller commands would be consistent.

Theorem 6. *$\mathcal{D}^{l*}_{F,M,Q}$ isolates all valid candidates.*

Proof. Assume an initial mode of fault occurrence $q_0 \in Q$, a candidate c with $|c| \leq l$, and a trace $\lambda = \sigma_1\sigma_2 \ldots \sigma_k \in L_{c,M,q_0}$. By the definition of a candidate trace, σ_1 is a candidate trace for $c' = (F_i, \mu(f, q_0))$ for $f \in F_i$ if $\sigma_1 \sqsubseteq \lambda' \in L_{f,M,\mu(f,q_0)}$. Therefore, $(f, \mu(f, q_0)) \in h_{F,M_i}(\sigma_1)$, so by definition of $\wedge_l$, the resultant diagnosis will contain $(f, \mu(f, q_0))$, so by definition of δ, the corresponding state is in S. Assume λ_i is a candidate trace for $c' = (F_i, q_i)$ and has a corresponding state $s \in S$. Then if $\sigma_{i+1} \in \Sigma_Q$, $\lambda_i\sigma_{i+1}$ is a candidate trace for $(F_i, \mu(\sigma_{i+1}, q_i))$ and by definition of δ has a corresponding state $s \in S$ and the associated diagnosis contains $(F_i, \mu(\sigma_{i+1}, q_i))$. If $\sigma_{i+1} \notin \Sigma_Q$, then $\lambda_i\sigma_{i+1}$ is a candidate trace for (F_i, q_i) if $\sigma_{i+1} \sqsubseteq \lambda' \in L_{f,M,\mu(f,q_0)}$ and therefore by definition of a hypothesis set, $(F_i, q_i) \in h_{F,M_i}(\sigma_{i+1})$, so by definition of $\wedge_l$, the diagnosis will contain (F_i, q_i) and by definition of δ, will have a corresponding state in S. Therefore, there is a state for any valid candidate trace. Given a state $s \in S$ with a trace that is maximal for $c = (F_i, q_i)$, the substate of s that corresponds to a state in $\mathcal{D}^*_{f,M,q_i}$, for some $f \in F_i$, must have no measurement deviations possible from its boundary, otherwise the trace would not be maximal, and thus the boundary must contain an accepting state. So, by definition of A, the state s is accepting. Since c, λ, and q_0 were general, $\mathcal{D}^{l*}_{F,M,Q}$ isolates all valid candidates by definition of isolation. □

Further, we can show that if the system $\mathcal{S}$ is diagnosable, then the diagnoser uniquely isolates all candidates.

Theorem 7. *A system $\mathcal{S} = (F, M, Q, L_{F,M,Q})$ is diagnosable if and only if $\mathcal{D}^{l*}_{F,M,Q}$ uniquely isolates all valid candidates.*

Proof. Assume $\mathcal{S}$ is l-fault diagnosable. Assume a c and $\lambda \in L_{c,M,Q}$, with $|c| \leq l$. $\mathcal{D}^{l*}_{F,M,Q}$ isolates c, so must have corresponding accepting state s with $c' \in Y(s)$ where c' covers c. Since $\mathcal{S}$ is diagnosable, there cannot be a c'' that does not cover c or that c does not cover where c and c'' are not distinguishable, by definition of l-fault diagnosability. So, there cannot be some common prefix λ that maps to an accepting state that has both a c' that covers c and a c''' that covers c''. So, $\mathcal{D}^{l*}_{F,M,Q}$ uniquely isolates all c, where $|c| \leq l$. Assume $\mathcal{D}^{l*}_{F,M,Q}$ uniquely isolates all c, where $|c| \leq l$. Then each possible fault trace λ has an accepting state s where $c' \in Y(s)$ and c' covers c. Thus, there cannot be some c'' that does not cover c or that c does not cover, with trace λ'' that reaches the same state, otherwise c'' or some candidate that covers it is in $Y(s)$. Therefore, c and c'' are distinguishable, so $\mathcal{S}$ is l-fault diagnosable. Thus $\mathcal{S}$ is l-fault diagnosable if and only if $\mathcal{D}^{l*}_{F,M,Q}$ uniquely isolates all c, where $|c| \leq l$. □

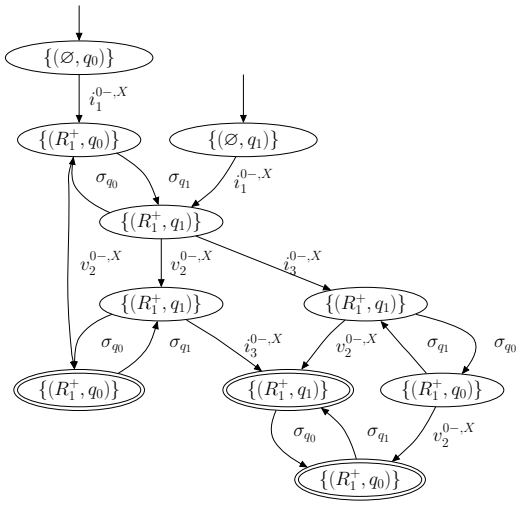


Figure 54: Hybrid diagnoser for R_1^+.

An implementation for the Π_l composition is given as Algorithm 14. First, it creates all the initial states. Since the mode events may now introduce cycles into the diagnoser, we include a visited states list, S_{visit}. The algorithm operates similar to that for MFD in continuous systems, except we now also have to perform the step of updating the hypotheses when a mode change event is received. When this happens, an operation analogous to `UpdateHypotheses` is performed. The new diagnosis contains all candidates in the previous diagnosis updated to the new mode via the mode event. Candidates which cannot be updated to the current mode because the mode change is not possible are eliminated. If the diagnosis is nonempty, a new state and new transition will be added.

The following example demonstrates how controlled mode changes affect the diagnoser design.

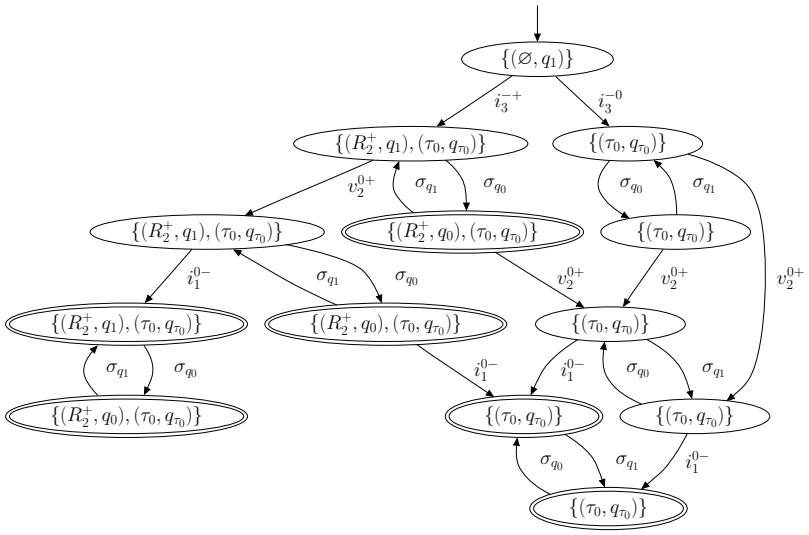
Example. Consider the R_1^+ fault. The diagnoser for $F = \{R_1^+\}$ only is given in Fig. 54. Both q_0 and q_1 may be initial expected modes at the time of fault detection. The mode transition function μ allows σ_{q_0} and σ_{q_1} to alternate. Starting in a particular mode, a measurement deviation indicates the occurrence of the fault. Either mode change events or further measurement deviations may occur. Consider the initial state given by the diagnosis $\{(\varnothing, q_0)\}$ and the trace $i_1^{0-,X} v_2^{0-,X}$. This trace is not a maximal candidate trace for the current candidate, $\{(R_1^+, q_1)\}$ because i_3 should still deviate. Therefore, the corresponding state is not accepting. However, if σ_{q_0} occurs next, then the new state is accepting, because in q_0, R_1^+ cannot effect i_3 since the switch is turned off, so the causal path is broken.

The following example illustrates the necessity of the discrete change feature in the fault signature to achieve diagnosability.

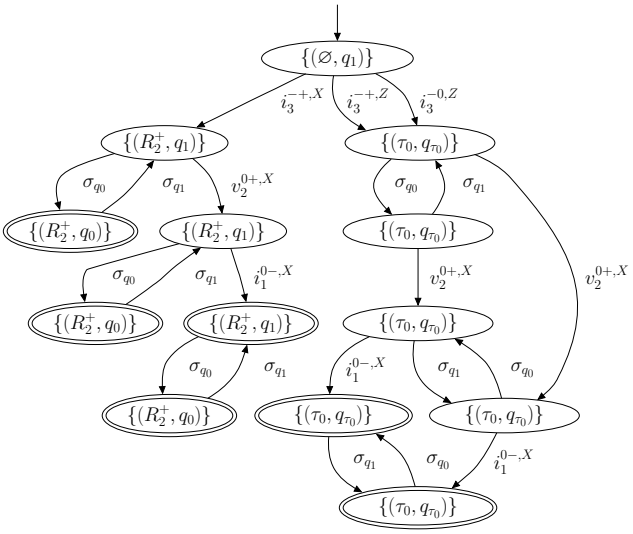
Algorithm 14 $\mathcal{D}^*_{F,M,Q} \leftarrow \Pi_l(\mathbb{D})$

$S \leftarrow \varnothing$, $I \leftarrow \varnothing$, $A \leftarrow \varnothing$, $\delta \leftarrow \varnothing$, $D \leftarrow \varnothing$, $S_{visit} \leftarrow \varnothing$
$\Sigma \leftarrow \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_k \cup \Sigma_Q$
for all $q \in Q_N$ **do**
  $s_0 \leftarrow (s_{0,1}, s_{0,2}, \ldots, s_{0,n}, q, (\varnothing, q))$
  $I \leftarrow I \cup \{s_0\}$
  $S \leftarrow S \cup \{s_0\}$
  $Y(s_0) \leftarrow (\varnothing, q)$
  $S_{pend} \leftarrow \{s_0\}$
  $S_{visit} \leftarrow S_{visit} \cup \{s_0\}$
while $S_{pend} \neq \varnothing$ **do**
  $s \leftarrow \texttt{pop}(S_{pend})$
  for all $\sigma \in \Sigma$ **do**
    if $\sigma \in \Sigma_Q$ **and** $s \notin I$ **then**
      for all $s_i \in s$ **do**
        $s'_i \in s' \leftarrow s_i$
      $q' \in s' \leftarrow \mu(\sigma, q)$
      $d \leftarrow \varnothing$
      for all $(f, q) \in Y(s)$ **do**
        if $\mu(\sigma, q) \neq \varnothing$ **then**
          $d \leftarrow d \cup (f, \mu(\sigma, q))$
    else
      $h \leftarrow \varnothing$
      for all $s_i \in s$ **do**
        if $\delta_{M(s)}(\sigma, s_i) \neq \varnothing$ **then**
          $s'_i \in s' \leftarrow \delta_{M(s)}(\sigma, s_i)$
          $h \leftarrow h \cup Y_i(s_i)$
        else
          $s'_i \in s' \leftarrow s_i$
      $q' \in s' \leftarrow q \in s$
      $d \leftarrow Y(s) \wedge_l h$
    if $d \neq \varnothing$ **then**
      if $s' \notin S_{visit}$ **then**
        $d' \in s' \leftarrow d$
        $\texttt{push}(S_{pend}, s')$
        $S_{visit} \leftarrow S_{visit} \cup \{s'\}$
        $S \leftarrow S \cup \{s'\}$
        for all $s'_i \in s'$ **do**
          if $\exists s_B \in B_{M(s')}(s'_i), s_B \in A_i$ **and** $c \in Y_i(s_B)$ covers some $c' \in Y(s')$ **then**
            $A \leftarrow A \cup \{s'\}$
            break
        $D \leftarrow D \cup \{d\}$
        $Y(s') \leftarrow d$
      $\delta(s, \sigma) \leftarrow s'$

Example. Consider the fault set $F = \{R_2^+, \tau_0\}$ with an initial mode of q_1. The diagnoser is shown in Fig. 55. Ignoring the discrete change feature (Fig. 55a), the system is not diagnosable, because there is at least one trace where the diagnoser ends up in an accepting state where the diagnosis contains multiple candidates. If we use the discrete change feature, then the system is unambiguously diagnosable (Fig. 55b). Note also that in Fig. 55, the hypothesized mode does not change for the discrete fault candidate when a mode change event is received, because once the discrete fault has occurred, the faulted switch is stuck so cannot change modes.



(a) Without the discrete change feature.



(b) With the discrete change feature.

Figure 55: Hybrid diagnosers for $F = \{R_2^+, \tau_0\}$ and initial mode of q_1 with $l = 1$.

As a particular case, consider the trace $i_3^{-+} v_2^{0+} \sigma_{q_0}$. Because R_2^+ cannot affect i_1 when in mode q_0, this is a maximal candidate trace for (R_2^+, q_0) and therefore corresponds to an accepting state. If it was τ_0 that had actually occurred, we would get a deviation in i_1, but since we do not know which fault will occur, we cannot guarantee that the trace will eventually distinguish a unique candidate. We would have to wait infinitely long to ensure that no further deviation occurs in order to verify that (R_2^+, q_0) is the true candidate.

The following example provides some diagnosability results for the circuit.

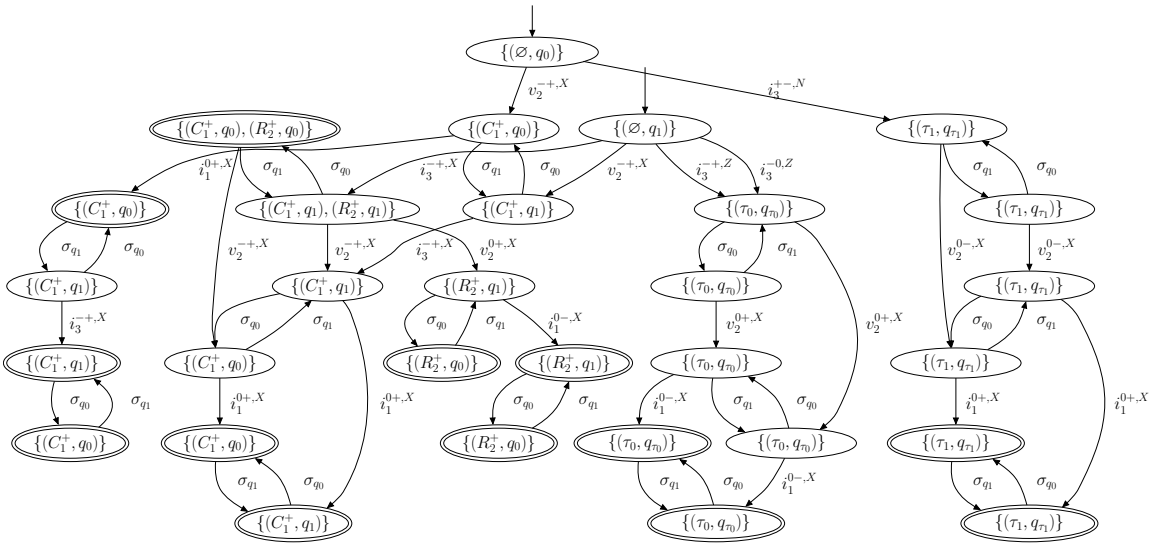


Figure 56: Hybrid diagnoser for $F = \{C_1^+, R_2^+, \tau_0, \tau_1\}$ with $l = 1$.

Example. For $F = \{C_1^+, R_2^+, \tau_0, \tau_1\}$, the hybrid diagnoser is given in Fig. 56. Although the system is 1-fault diagnosable in the continuous case, the diagnoser shows that the system is not 1-fault diagnosable in the hybrid case. Starting in q_1, if the trace $i_3^{-+,X}\sigma_{q_0}$ occurs, both C_1^+ and R_2^+ are consistent. Since R_2^+ cannot affect the measurements in q_0, the trace is maximal for this candidate, so corresponds to an accepting state. Since we do not know which fault occurred and the diagnoser has no control over controlled mode changes, we cannot guarantee that the true candidate will eventually be distinguished. However, the system is 1-fault Q-diagnosable, because we can always either leave the state by executing controlled mode changes or prevent entrance of the state by blocking controlled mode changes. From this state, if we issue σ_{q_1}, we enter a state from which the candidates can be distinguished. Once in that state, how v_2 deviates will determine the true candidate, as long as we prevent σ_{q_0} from occurring until we see v_2 deviate. Alternatively, we can prevent σ_{q_0} from occurring after $i_3^{-+,X}$ so that we never enter the state, thus ensuring the a deviation from v_2 will be the next event.

The following examples demonstrate the operation of the hybrid diagnoser.

Example. Consider the diagnoser for $F = \{R_2^+, \tau_0, \tau_1\}$ with $l = 2$ and initial mode q_0, shown in Fig. 57, and the trace $i_3^{+-,N} v_2^{0+,X} i_1^{0-,X}$. After $i_3^{+-,N}$ the diagnosis is $\{(\tau_1, q_{\tau_1})\}$. After $v_2^{0+,X}$, the hypothesis set is $\{(R_2^+, q_1), (R_2^+, q_{\tau_1}), (\tau_0, q_{\tau_0})\}$, so the diagnosis is $\{(R_2^+\tau_1, q_{\tau_1})\}$. Candidate $(\tau_0\tau_1, q_{\tau_0})$ is not included because q_{τ_0} is not possible from q_{τ_1}, and it violates the persistency assumption, since the two fault events correspond to the same switching component. After $i_1^{0-,X}$, the hypothesis set is $\{(R_2^+, q_1), (R_2^+, q_{\tau_1})\}$, so the diagnosis remains $\{(R_2^+\tau_1, q_{\tau_1})\}$.

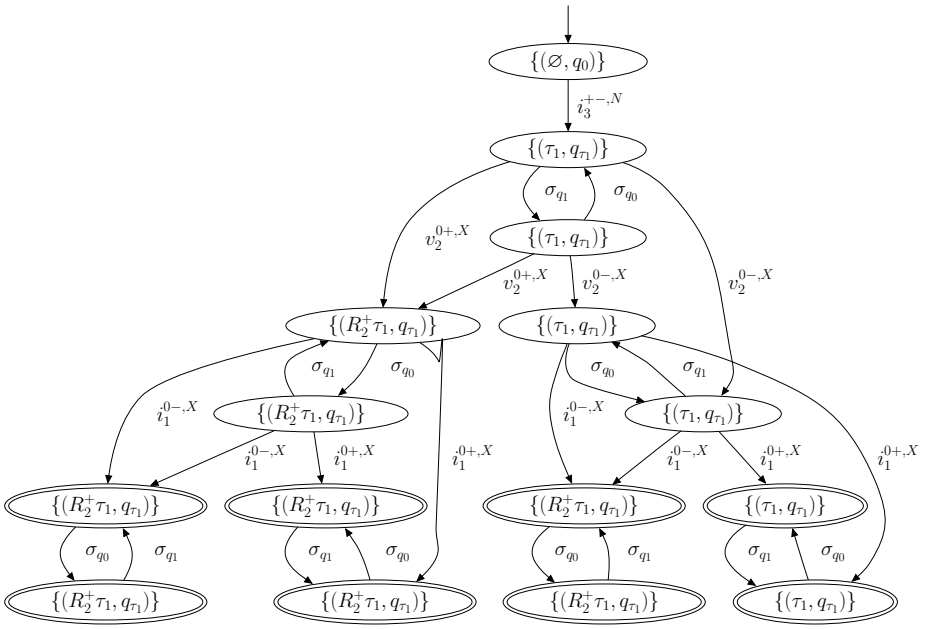


Figure 57: Diagnoser for $F = \{R_2^+, \tau_0, \tau_1\}$ with $l = 2$ and initial mode q_0.

Consider the diagnoser for $F = \{R_2^+, \tau_0, \tau_1\}$ with $l = 2$ and initial mode q_1, shown in Fig. 58. Consider the trace $i_3^{-+,X} v_2^{0+,X} i_1^{0-,X}$. After $i_3^{-+,X}$, the hypothesis set contains only (R_2^+, q_1), so the diagnosis becomes $\{(R_2^+, q_1)\}$. After $v_2^{0+,X}$, the hypothesis set is $\{(R_2^+, q_1), (R_2^+, q_{\tau_1}), (\tau_0, q_{\tau_0})\}$, so the diagnosis becomes $\{(R_2^+, q_1), (R_2^+\tau_0, q_{\tau_0})\}$, but is pruned to $\{(R_2^+, q_1)\}$ to make it minimal. After $i_1^{0-,X}$ the hypothesis set is again $\{(R_2^+, q_1), (R_2^+, q_{\tau_1}), (\tau_0, q_{\tau_0})\}$, so the diagnosis does not change.

Consider the diagnoser for $F = \{R_2^+, \tau_0, \tau_1\}$ with $l = 2$ and initial mode q_1, shown in Fig. 58, and the trace $i_3^{-+,Z} v_2^{0+,X} \sigma_{q_0} i_1^{0-,X}$. After $i_3^{-+,Z}$, the hypothesis set contains only (τ_0, q_{τ_0}), so the diagnosis becomes $\{(\tau_0, q_{\tau_0})\}$. After $v_2^{0+,X}$, the hypothesis set is $\{(R_2^+, q_1), (R_2^+, q_{\tau_1}), (\tau_0, q_{\tau_0})\}$ and the diagnosis remains the same. After σ_{q_0}, an actual mode change does not occur because q_0 is not reachable from q_{τ_0}, so the diagnosis remains the same. After $i_1^{0-,X}$, the hypothesis set is $\{(R_2^+, q_1), (R_2^+, q_{\tau_1}), (\tau_0, q_{\tau_0})\}$ and the diagnosis remains the same.

Online Implementations

The proposed event-based diagnosis framework leads to three different implementations of the event-based diagnoser that trade off space and time complexity. In general, the more information that is compiled, the more efficient the diagnoser is, but the more space it requires. If less information is compiled, the diagnoser requires less space, but becomes less efficient.

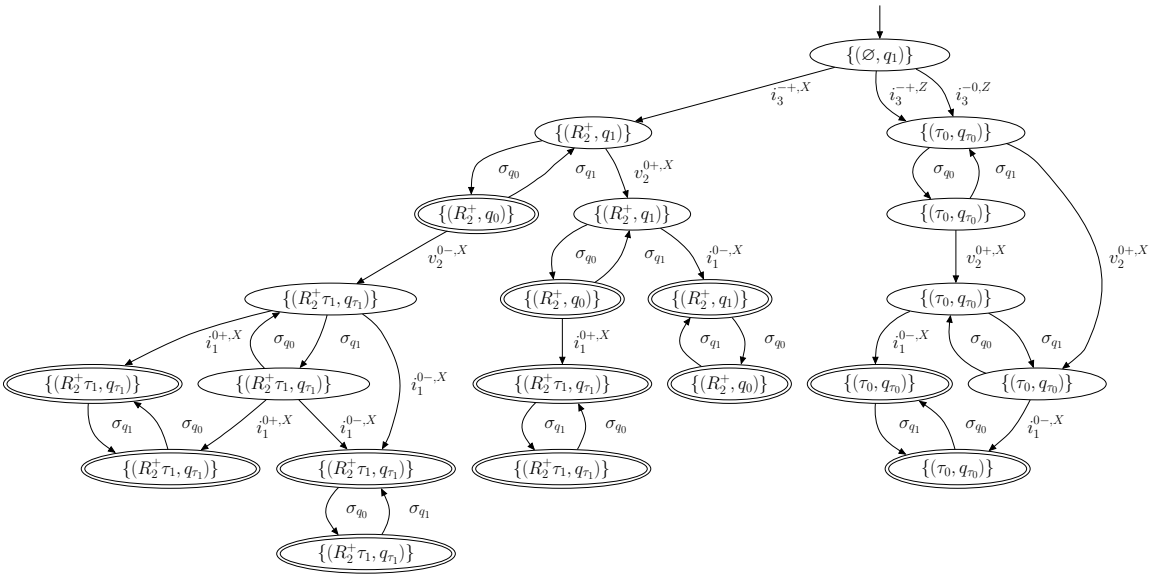


Figure 58: Diagnoser for $F = \{R_2^+, \tau_0, \tau_1\}$ with $l = 2$ and initial mode q_1.

The Global Diagnoser Implementation

Performing online diagnosis with the global diagnoser, $\mathcal{D}^{l*}_{F,M,Q}$, has the best time complexity. At design time, the global diagnoser is constructed based on the fault models. At run-time, all possibilities are captured by the diagnoser, so the diagnoser needs only to wait for events to occur, transition to the next state, and output the associated diagnosis. Using appropriate data structures, these operations can be achieved in constant time.

With a large number of faults and measurements, however, the global diagnoser may have poor space complexity. Since it must contain all the possible event sequences, it must capture $O(|F|T)$ traces, where T is the maximum number of traces per fault. This is also the design-time complexity of constructing the global diagnoser. In the worst case, a fault may have no measurement orderings, thus T is $O((|M|+|Q|)!)$. Therefore, there would be $O(|F|(|M|+|Q|)!)$ traces and $O(|F|(|M|+|Q|)!)$ states in the worst case. If T is truly the worst case, however, i.e., a fault allows all possible event sequences, then the diagnoser would only have $O((|M| + |Q|)!)$ distinct traces to capture, and thus $O((|M| + |Q|)!)$ states.

In SFD for continuous systems, the global diagnoser needs only to represent at most $O(|F|T)$ traces, so if many temporal orderings exist, then the number of possible fault traces reduces significantly, and the global diagnoser will have feasible space requirements. In MFD for continuous systems, the number of traces increases, because the diagnoser has to represent all the traces of the single faults as well as the additional traces that may arise when the faults occur together, so

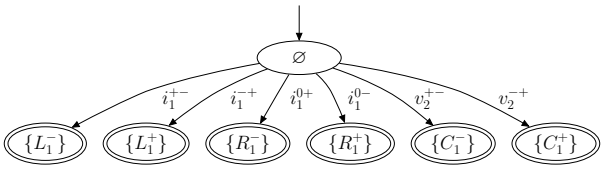


Figure 59: Pruned diagnoser for the circuit for mode q_0 with $l = 1$.

space requirements increase. Depending on the selection of l, space requirements may or may not be feasible. In hybrid diagnosis, mode change events are included, and all possible mode changes have to be accounted for in order to establish diagnosability of the system. If there are a large number of modes, however, the space requirements for the global diagnoser become a limiting factor. The diagnoser would be the most time-efficient, but its space-efficiency would explode exponentially.

Space requirements for the global diagnoser can be reduced by pruning of the states. The diagnoser can be pruned by recursively removing leaf states, i.e., states from which no further measurement deviation events are possible (but controlled mode changes still might be) that have the same diagnosis (independent of the mode) as their parent states. Since removal of leaf states may create more leaf states, the procedure is recursive. For example, Figs. 59 and 60 show the pruned diagnosers for modes q_0 and q_1 of the circuit with $l = 1$, respectively. In many cases, diagnosers with $l > 1$ will not be pruned very much, because it is likely that new events may expand the diagnosis, whereas with single faults, the diagnosis can only reduce as more events occur.

When pruning diagnosers, newly created leaf states are made accepting states. Therefore, accepting states no longer exclusively correspond to maximal traces, but also to traces for which the diagnosis can no longer change. Therefore, the leaf states correspond to termination points. When a leaf state is reached in a pruned diagnoser, it is known that the faults in the candidates of the diagnosis cannot change for any future measurement deviations or controlled mode changes. So, fault isolation can be terminated, knowing that the best result has been achieved. Therefore the pruned global diagnoser provides us with exact points at which fault isolation can safely terminate, without the use of time limits as discussed in Chapters V and VI.

Another way to reduce the space requirements of the global diagnoser is to employ a distributed approach, where diagnosers for subsystems have only a subset of the faults and measurements. This approach will be described in the case study of Chapter VIII within the context of single fault diagnosis in continuous systems. Extension of the distributed approach to multiple faults and hybrid systems is left as future work.

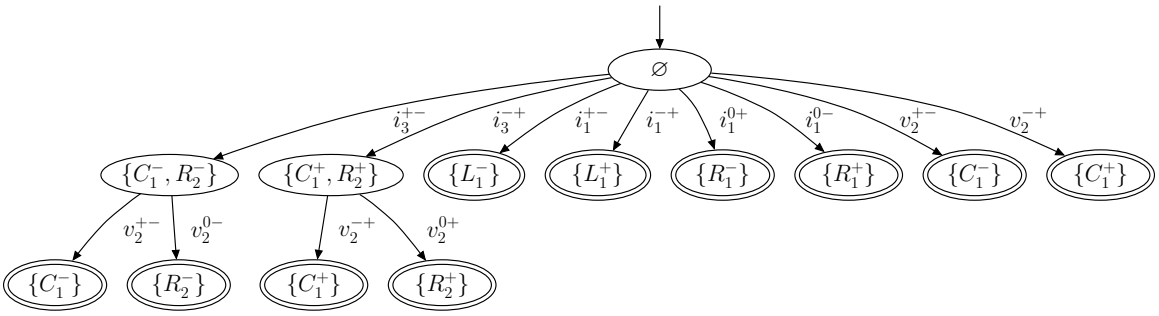


Figure 60: Pruned diagnoser for the circuit for mode q_1 with $l = 1$.

The Subdiagnoser Implementation

In the subdiagnoser implementation, only the individual $\mathcal{D}^*_{\{f\},M,q}$ for each $f \in F$ and $q \in Q$ are computed at design time, which is less expensive than computing the global diagnoser. Each fault may have, in the worst case, $O(|M|!)$ possible fault traces. The worst case total space requirement is then $O(|F||Q||M|!)$, because each fault-mode pair may have $O(|M|!)$ traces to capture. Again, if many temporal orderings exist, then the space complexity may reduce substantially.

In SFD for continuous systems, the global diagnoser will always take up less space, since it merges common fault traces and prefixes from the individual fault models [86, 133]. For MFD for continuous systems, the subdiagnoser approach will in most cases have better space complexity than the global diagnoser, because the additional traces that arise when multiple faults occur are not explicitly represented, but rather, computed online. In hybrid systems diagnosis, the subdiagnoser approach can be implemented by generating the necessary subdiagnosers online for the hypothesized modes. Therefore, only the subdiagnosers that are needed are computed.

In online diagnosis, each subdiagnoser is traced simultaneously. The hypothesis set, h, is formed when new events occur, as described in Chapter V for continuous systems and Chapter VI for hybrid systems. This operation has time complexity $O(|F||Q|)$, since it must check at worst each fault model for consistency. The current diagnosis is formed using the $\wedge_l$ operator, which is a function of the size of the hypothesis set and the size of the current diagnosis. At worst, the hypothesis set contains $O(|F||Q|)$ elements, and the diagnosis contains $O((|F||Q|)^l)$ elements, so the $\wedge_l$ operation takes $O((|F||Q|)^l)$ time. In practice, time complexity is reduced because diagnoses are pruned to be minimal, so the size of the diagnosis stays small [39]. The online composition of the fault models corresponds to constructing the *path* of the diagnoser for the particular fault trace observed. Single fault information along with controlled mode change information is composed online to obtain the interleaved traces, so we do not need to pre-compute all possible traces.

The Diagnoser-free Implementation

If each fault has many measurement orderings, then using either the global diagnoser or subdiagnoser approach may be both space-efficient and time-efficient for continuous systems. If few orderings are available, then the diagnosers approach size $O(|M+Q|!)$, therefore, these approaches may not be feasible given the space requirements of the system. The third implementation computes only the fault signatures and relative measurement orderings for each fault online when fault isolation begins. This case corresponds to directly using the fault isolation algorithms of Chapter V and VI online.

In diagnosis with this approach, we form the hypothesis set corresponding to the current measurement deviation by looking through the fault signatures and measurement orderings, and this requires $O(|F||Q||M|^2)$ time in the worst case. We then compute the new diagnosis, which takes $O((|F||Q|)^l)$ time. Therefore, this approach has the worst time complexity because nothing is compiled at design time. Everything is computed at run-time, so the space requirements are minimal, because only the relevant path of the global diagnoser is being constructed online, and signatures and orderings are only computed as needed. Therefore, for hybrid systems, this is the best solution for systems with a large number of modes, since the global diagnoser becomes very large when the number of modes increases. This also allows us to exploit one of the main advantages of hybrid bond graphs, i.e., avoiding pre-enumeration of system modes. However, in order to verify diagnosability properties, the global diagnoser must still be computed.

Summary

Our event-based approach to diagnoser design can be viewed as a compilation strategy. Offline, event-based fault models are converted into diagnosers, which are then composed into global diagnosers. The global diagnosers are very time-efficient, although for multiple faults and especially for hybrid systems, can have very large space requirements. The diagnosers provide diagnosability analysis of systems, and show whether unique isolation of faults can be achieved, and if not, what particular traces yield ambiguous diagnosis results. For hybrid systems, they also show which controlled mode changes should be prevented or executed during fault isolation to avoid ambiguities when possible. Because of the sometimes poor space complexity of the global diagnosers, a spectrum of diagnoser implementations is presented that ranges from no compilation to full compilation, and the trade-offs between the approaches were discussed.

Though diagnosis using ordered event sequences is performed similarly in DES approaches and our approach, the main contrast between them is the abstraction used to generate the DES models.

Most of the traditional DES approaches assume models created by human experts, so the DES model and all its faulty behavior is assumed to be given [5, 6, 8, 62–66]. As opposed to quantization-based abstractions [68, 69], we perform a qualitative abstraction with respect to nominal behavior, and therefore the quality of the abstraction is not affected by fault magnitude or time of fault occurrence. Measurement deviations directly indicate the presence of a fault, so our diagnosers track only faulty behavior, whereas diagnosers in DES approaches also track the nominal behavior (e.g., see Fig. 4 of Chapter II). The individual fault diagnosers generated by our approach are similar to chronicles and the specification diagnosers of [63]. The global diagnoser bears resemblance to those in [5, 6]. Our approach is also similar to timed failure propagation graphs (TFPGs), which include additional information such as time bounds on symptom appearance [101, 135, 136]. Since time bounds are used, diagnosability is, in general, increased, although how to systematically generate a TFPG for a dynamic system is not addressed. Our approach represents one way to generate the some of the information needed for a TFPG for a dynamic system.

Because we are working in an event-based framework, the notions of fault traces, fault languages, distinguishability, and diagnosability bear a resemblance to those defined in the DES framework. The notion of a fault trace is similar to the notion of a fault signature defined for DES in [137], where it is defined as a string (of finite or infinite length) that contains a fault event. Diagnosability can then be defined in terms of ensuring no two faults have the same signature (in our case, the same trace). This is the situation in any modeling framework, because in general, faults can only be distinguished if they manifest in different ways, in whatever way that is represented by the model. Our approach separates out the fault effects and analyzes them separately.

Our notion of distinguishability is equivalent to that of strong discriminability in [134], and, therefore, our notion of diagnosability is similar to strong diagnosability in [134], except we establish diagnosability only in terms of minimal candidates. Single faults are treated the same as multiple faults in the definition of [134], so with fault masking, diagnosability will almost never be achieved under that definition. Also, no event-based information is used in [134]. Our definition of diagnosability for hybrid systems is also similar to the approach presented in [138], because events in the form of mode changes are taken into account. The approach uses analytical redundancy relations but does not use any temporal orderings between symptom appearance as in the ARR approach of [25, 26], therefore our formulation of diagnosability is more general.

CHAPTER VIII

CASE STUDY: DISTRIBUTED DIAGNOSIS IN MOBILE ROBOT FORMATIONS

Autonomous multi-robot teams can perform a wide range of collaborative tasks in manufacturing, surveillance, and space exploration. In many cases, the execution of the task requires formation control [139–142], and the success of the overall operation depends on each robot operating in an error-free manner. Faults in one robot can propagate to other robots over communication links, and this can cause problems in maintaining the formation required to execute the desired tasks (e.g., collaboratively moving a load [95]). Degradations and faults must be detected and isolated early to allow for reconfiguration and continued operation, and this can be achieved only if diagnostic mechanisms are incorporated into multi-robot systems [143]. However, the diagnosis of robot formations is a difficult problem. A global, centralized model is usually needed to capture the interactions that govern the propagation of fault effects between robots, but centralized approaches have many weaknesses. Specifically, these approaches (*i*) create a single point of failure, (*ii*) do not scale well as formation size increases, (*iii*) do not exploit the computational resources available on each robot, and (*iv*) incur large communication overhead.

Due to the problems with centralized approaches, we employ our diagnosis approach in a distributed fashion to formations of mobile robots, where each robot has a local diagnoser. Most work in diagnosis of mobile robots has concentrated on the single-robot case. A survey of such methods can be found in [144]. TRANSCEND is applied in the single-robot case for diagnosis of actuator faults using fault signatures derived from a simplified bond graph model [145]. The parity relation approach is applied to nonlinear single-robot systems in [93]. Particle filtering techniques are employed in single-robot diagnosis also [31,146]. Fault detection in mobile robots is addressed in by developing a technique which accounts for both kinematic and dynamic behaviors in order to generate better residuals in spite of parametric uncertainty [147]. Sensor fault detection and identification using multiple model adaptive estimation based on a bank of Kalman filters is developed in [148], and extended by using a neural network to detect and identify both sensor and mechanical failures based on the output of the filter bank in [149].

Multi-agent and distributed diagnosis have been explored previously as well, but has not been directly applied to mobile robots. Distributed systems are diagnosed using an agent framework where some failures are diagnosed locally, and others require coordination between the agents in [150]. Local

diagnosers construct local diagnoses such that they are consistent with global diagnoses, sacrificing diagnostic precision for gains in computational complexity in [151]. The problem of addressing coordination failures in multi-robot teams is addressed in [152, 153]. Our approach deals with process faults, whereas coordination failures are better described as logical faults, which are at a higher level. Such approaches, however, can be considered as complementary to our work.

In this case study, we apply our diagnosis methods to single fault diagnosis in mobile robot formations, where the formations can be modeled as continuous systems. Due to the problems with a centralized approach, we apply the distributed TRANSCEND methodology [40], but extend it to incorporate relative measurement orderings. The distributed approach enables each robot to, using a local diagnoser, individually determine a globally correct local diagnosis with a small set of measurements, therefore avoiding the need for a centralized diagnosis coordinator. Our approach employs an extended bond graph model that comprehensively models the physical components, sensors, and actuators, as well as the communication processes among the robots. Interactions between the robots cause fault effects to propagate across robot boundaries, and, therefore, require additional discriminatory power to isolate all the faults of interest. Relative measurement orderings are key to solving this problem [28, 95]. A formal diagnosability analysis for single, persistent faults in robot formations shows that a combination of fault signatures and relative measurement orderings increases the discriminatory power of the measurements and facilitates more efficient diagnoses. In contrast to a centralized diagnosis approach, our solution scales well to large formations, minimizes the communication costs associated with fault isolation, takes advantage of the computational resources available on each robot, and avoids the need for a centralized coordinator for the local diagnoses. Experimental results a four-robot formation system demonstrates the effectiveness of this approach. The results illustrate the advantages of the method, namely (*i*) scalability, (*ii*) increasing the discriminatory power of the measurements, and (*iii*) improving the efficiency of the distributed diagnosis approach (see also [28, 95]).

Our approach is applicable to rigid formations of mobile robots. In rigid formations there is a strong coupling between the dynamics of the robot behaviors, which is exploited by our algorithm to improve the discriminatory power of the measurements and the efficiency of the diagnosers. The approach may not offer advantages for collaborative multi-robot applications that do not exhibit coupling between the dynamic behavior of individual robots. Specifically, for applications where faults do not propagate between the mobile robots, our method can be still applied but will naturally result in independent diagnosers for each robot.

The chapter is organized as follows. We first present our modeling methodology. We then describe the multi-robot diagnosis problem, and presents the computational architecture of the diagnosis scheme. We then present the theoretical results in applying our approach to formation diagnosis. We next demonstrate the effectiveness of the approach using experimental results for a four-robot formation, followed by a summary of the case study.

Modeling Formations of Mobile Robots

In this case study, we develop diagnostic solutions for formations of robots that employ leader-based control schemes of the type described in [139, 140]. The approach assumes that there exists a single global leader, for the purposes of control, who follows a known, planned trajectory. The remaining robots in the formation, i.e., the followers, must maintain their positions with respect to the global leader and/or other followers (local leaders). Correct behavior is defined with respect to a global objective. The global objective of the system is to maintain the overall formation while pursuing the planned trajectory and executing predefined tasks, e.g., collecting information or pushing a load.

Each follower robot implements two control laws, governing its translational and rotational velocities. These laws are functions of a robot's local information and information generated by its leaders. Therefore, the approach is scalable to a large number of robots without a corresponding increase in algorithm complexity. The formation model of [139, 140] is also general enough to model any type of rigid formation, since each robot's position is defined with respect to other robots in the formation. We adopt this formation modeling and control approach in our work, and apply our diagnosis framework to these robot configurations.

The control algorithms in [139,140] assume a kinematic model of the robot, given the translational and rotational velocities as inputs. We develop a bond graph model of the system that captures both the kinematic and dynamic behavior of the robot under nominal and faulty system operation. The formation control mechanisms are explicitly built into the bond graph model for diagnosis, and, therefore, our approach can be used with any control scheme that ensures the robot team is in a rigid formation.

Each robot includes a local controller that regulates the velocities of its two wheels. The sensor suite includes motor encoders to measure wheel velocity and a gyroscope to measure heading. A distributed controller coordinates the formation by determining the desired velocities for each robot based on local and remote sensor measurements, communicated via a wireless network. In the remainder of the section, we present the model of the multi-robot system used for diagnosis.

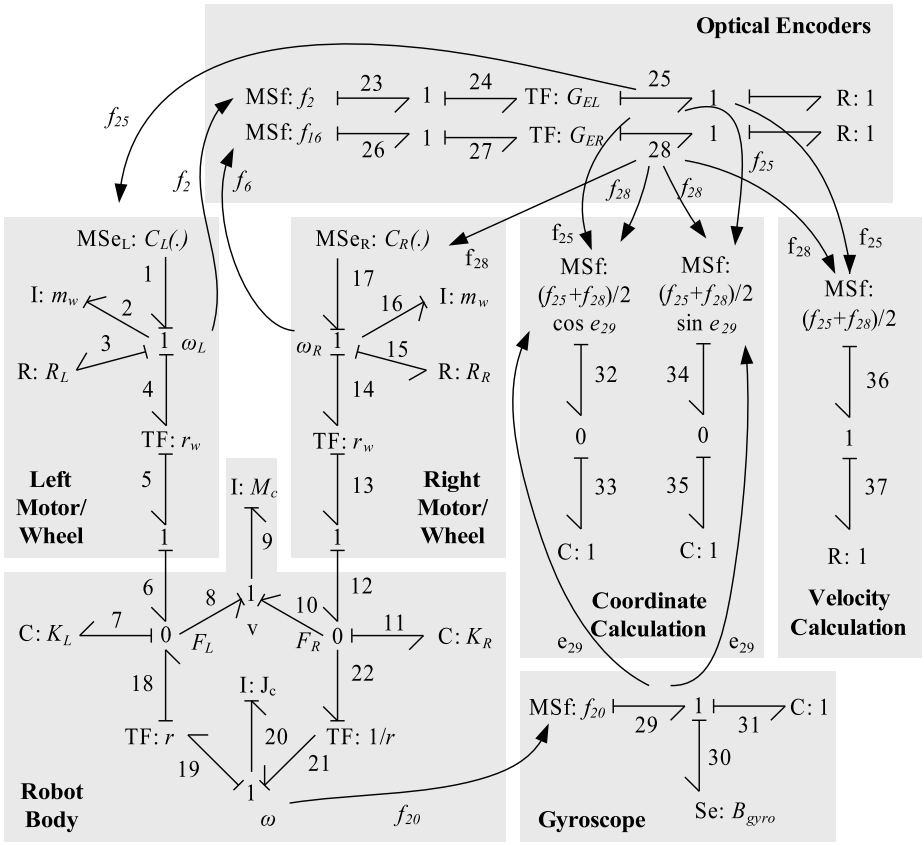


Figure 61: Bond graph model of a single robot.

Modeling a Single Mobile Robot

Each robot is modeled using a bond graph. The bond graph for a single robot is shown in Fig. 61. A single robot consists of left and right wheel drive subsystems, a chassis, a gyroscope, and two motor encoders. The 1-junctions represent the common velocity points, e.g., the rotational velocity of the left wheel, ω_L, the rotational velocity of the right wheel, ω_R, the forward velocity of the robot, v, and the rotational velocity of the robot, ω. The 0-junctions represent common force points, e.g., the forces on the left and right sides of the robot, F_L and F_R. The wheel subsystems include modulated sources of effort (MSe_L and MSe_R) that model the actuator torque outputs that directly feed the wheels, and inertial elements that model wheel mass and inertia, m_w. Resistive elements (with parameters R_L and R_R) model energy dissipation (i.e., friction) in the wheels. Transformers model the transformations between linear and rotational velocities. The robot body subsystem includes inertia components that model robot mass, M_c, and rotational inertia, J_c. The capacitive components (with parameters K_L and K_R) model the mechanical stiffness of the robot system.

Sensor models in the bond graph are derived from the kinematic relationships between the robot velocities and the measurements. Each robot includes a gyroscope and two motor encoders. The gyroscope computes the heading, $\theta(t)$, using a kinematic equation based on the rotational velocity, $\omega(t)$, of the robot body, i.e., $\dot{\theta}(t) = \omega$. The equations for the optical encoder measurements involve a gain transforming the wheels' rotational velocities to their linear velocities, i.e., $v_L(t) = G_{EL}\ \omega_L(t)$ and $v_R(t) = G_{ER}\ \omega_R(t)$, where G_{EL} and G_{ER} are the encoder gains for the left and right wheels, respectively. These are used to calculate the measured translational velocity, v, of the robot. In the bond graph of Fig. 61, v is represented by the flow variable f_{36}, associated with bond 36.

The sensors are modeled in the bond graph as modulated sources of flow that encapsulate the measurement equations for v_L, v_R, and θ. For the gyroscope, the flow source is the rotational velocity of the robot, ω, represented in the model as f_{20}. The measured variable, the heading, is e_{29} (the effort variable associated with bond 29), which is the integral of ω plus the sensor bias (if any). For the case of the optical encoders, the flow is the rotational velocity of a wheel (ω_L and ω_R) passed through a gain, so the measured variables are f_{25} and f_{28}.

Position information is calculated in the bond graph using velocity and heading information. These are also modeled using modulated sources of flow. The x and y coordinates are described by:

$$\begin{aligned} \dot{x}(t) &= \frac{v_L(t) + v_R(t)}{2} \cos\theta(t), \\ \dot{y}(t) &= \frac{v_L(t) + v_R(t)}{2} \sin\theta(t). \end{aligned}$$

The modulated sources of flow provide these quantities, which are integrated to obtain the coordinate positions of the robot, e_{33} for the x coordinate, and e_{35} for the y coordinate.

Local controllers are also modeled in the bond graph. The input to the robots are the motor torques modeled as modulated sources of effort, which encapsulate the wheel control equations. In our model, each wheel has an accompanying PID controller. For example, the equation of the controller for the left wheel is given by:

$$\tau = K_p(v_L - v_{Ld}) + K_i \int_0^t (v_L - v_{Ld})dt + K_d \frac{d(v_L - v_{Ld})}{dt},$$

where τ is the torque applied by the motor, K_p, K_i, and K_d are the controller gains, and v_{Ld} is the reference velocity provided by the formation controller described in the next subsection. The torque for the left (right) wheel is represented in the bond graph by the modulated source of effort MSe_L (MSe_R). The PID controller is represented in the bond graph by the function $C_L(\cdot)$ ($C_R(\cdot)$) that

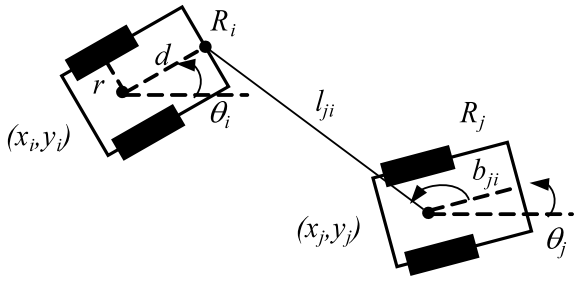


Figure 62: Control variables in robot formations.

modulates the torque. The edges from the observed velocities to the wheel sources represent the control links for the PID controllers. Other controller types can be modeled similarly.

Modeling Leader-based Formations

Following the approach in [139], we model a formation as a tuple $\mathcal{F} = (\mathcal{S}, \mathcal{C})$, where $\mathcal{S}$ is a set of shape variables defining the formation structure, and $\mathcal{C}$ is a control graph showing the control strategies for each robot and dependencies on their neighbors. The shape variables $\mathcal{S}$ consist of relative bearings and separations between robots. Control laws maintain either the relative heading and separation of a follower to its leader (separation-bearing control, or SBC), or the separations of a follower from two leaders (separation-separation control, or SSC). In this way, formations can be constructed by defining for each robot its control strategy, shape variables, and leaders.

Definition 54 (Control Graph)**.** A control graph $\mathcal{C}$ is a directed, acyclic graph, where each robot, R_i, defines a vertex. A directed edge (R_i, R_j) implies that R_i is a local leader to R_j, i.e., R_j maintains its position with respect to R_i.

As in [139], we restrict a control graph with the following constraints: (i) The formation leader, R_1, has no incoming edges and at least one outgoing edge, and (ii) all other robots have at least one and no more than two incoming edges. If a robot has exactly one incoming edge, then it employs the SBC strategy, otherwise it has two incoming edges and it employs the SSC strategy. In general, a robot with three or more incoming edges is over-constrained for planar formations, so this is disallowed.

We denote by l_{ji} the separation between R_j and R_i and by b_{ji} the relative bearing from R_j to R_i as measured from R_i's axis of symmetry to the line connecting the center of R_j's wheel axis with the point d units from the center of R_i's wheel axis, as shown in Fig. 62. A subscript d denotes a desired value, and a subscript i indicates a variable associated with R_i.

With local leader R_j, the SBC control equations [140] for the follower, R_i, describe the desired rotational velocity ω_{di} and linear velocity v_{di}, and are given as follows.

$$\begin{aligned}\omega_{di} &= \frac{\cos\gamma_j}{d}\{\alpha_b l_{ji}(b_{dji}-b_{ji}) - v_j \sin b_{ji} + l_{ji}\omega_j + \\ &\qquad \rho_{ji}\sin\gamma_j\} \\ v_{di} &= \rho_{ji} - d\omega_{di}\tan\gamma_j,\end{aligned}$$

where

$$\begin{aligned}\rho_{ji} &= \frac{\alpha_l(l_{dji}-l_{ji}) + v_j\cos b_{ji}}{\cos\gamma_j}, \\ \gamma_j &= \theta_j + b_{ji} - \theta_i,\end{aligned}$$

and α_l and α_b are control gains for the separation and bearing, respectively.

With local leaders R_j and R_k, the SSC control equations [140] for the follower, R_i, are given as follows.

$$\begin{aligned}\omega_{di} &= \frac{1}{d\sin(\gamma_j-\gamma_k)}\{\alpha_{lj}(l_{dji}-l_{ji})\cos\gamma_k + \\ &\qquad v_j\cos b_{ji}\cos\gamma_k - \\ &\qquad \alpha_{lk}(l_{dki}-l_{ki})\cos\gamma_j - \\ &\qquad v_k\cos b_{ki}\cos\gamma_j\} \\ v_{di} &= \frac{\alpha_{lj}(l_{dji}-l_{ji}) + v_j\cos b_{ji} - d\omega_{di}\sin\gamma_j}{\cos\gamma_j},\end{aligned}$$

where α_{lj} and α_{lk} are control gains for separations from R_j and R_k, respectively.

The formation control outputs v_{di} and ω_{di} can be decoupled into individual left and right wheel velocities to serve as inputs to the wheels' PID controllers as $v_{Ldi} = v_{di} - r\omega_{di}$ and $v_{Rdi} = v_{di} + r\omega_{di}$, where r is half the wheel base (see Fig. 62).

Fig. 63 shows an example formation structure. R_1 is the leader, and moves in a pre-defined trajectory. R_2 employs SBC, so it only maintains its separation and bearing with respect to R_1. R_3 uses SSC, maintaining its separation from R_1 and R_2. R_5 and R_6 also utilize SSC, and R_4 uses SBC.

The distributed control algorithm is modeled in the bond graph in the same manner as the local PID control. Because heading, position, and velocity measurements are required for the control,

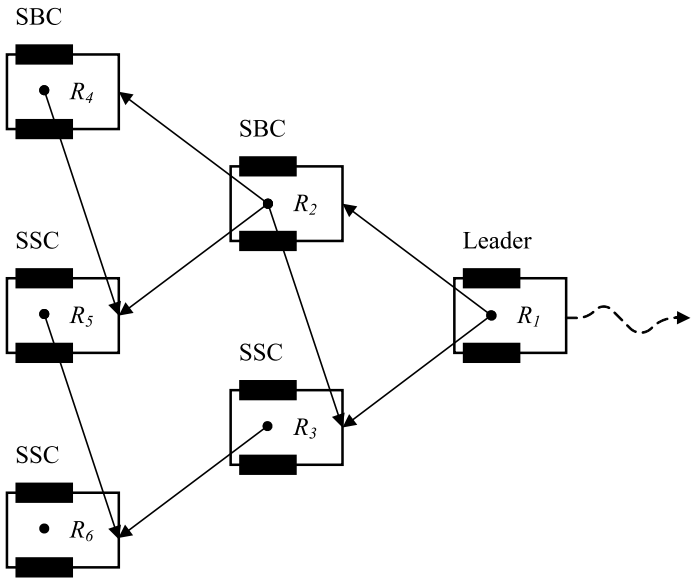


Figure 63: Example formation of six robots.

signals are introduced from each robot's x, y, θ, and v measurements to its own wheel sources, and also to the wheel sources of each follower to represent the communication between the robots. For example, following the formation in Fig. 63, signal edges are constructed from v_2 ($f_{36,2}$) of R_2 to $MSe_{L,3}$ ($e_{1,3}$) and $MSe_{R,3}$ ($e_{17,3}$) of R_3 because $C_{L,3}(\cdot)$ and $C_{R,3}(\cdot)$ take v_2 as an argument. The multi-robot bond graph is derived from the composition of single robot bond graphs with these signal edges included.

Modeling Faults

We consider a set of parametric faults in the actuators and sensors for the robots, shown in Table 7 with the corresponding parameters in the bond graph model (a superscript of + or − indicates the direction of change of the parameter value). Actuator (motor) faults are modeled as changes in the effort sources. A saturation fault in an actuator limits the maximum wheel velocity. Sensor bias is modeled as an additive fault, and is represented by a change in the effort source at the measured value (nominally the effort is 0). For example, a bias in the gyroscope manifests as an abrupt, constant value added to the true measurement value. Sensor failures are modeled as multiplicative faults and are parameterized by a change in the sensors' transformer gains. For the optical encoders, the nominal value of G_{EL} (or G_{ER}) is r_w, the wheel radius. A fault in the encoder is modeled as a reduction in gain, i.e., its value reduces to a number in the interval $[0, r_w)$. This corresponds to a percentage of the encoder counts that are missed (at least 10%).

Table 7: Fault Parameters in the Robot Models

Fault	Description
MSe_L^-	Left actuator saturation/failure
MSe_R^-	Right actuator saturation/failure
G_{EL}^-	Left encoder (partial) failure
G_{ER}^-	Right encoder (partial) failure
B_{gyro}^+	Positive gyroscope bias
B_{gyro}^-	Negative gyroscope bias

Problem Formulation

If a system is globally diagnosable, then a centralized diagnoser can be constructed that can uniquely isolate all faults, as shown in Chapter VII. Such an approach, however, results in a very large diagnoser that becomes a single point of failure. The single point of failure can be avoided by replicating the centralized diagnoser on each robot, however, this will be inefficient for large formations. In addition, the diagnosers on each robot will perform unnecessary computations involving fault hypotheses that are not relevant to the particular robot. We instead take a distributed approach, where each local diagnoser isolates faults in its subsystem using local measurements and some remote measurements, if required. Since accessing remote measurements is expensive, our design goal is to find the minimum number of remote measurements that makes each subsystem globally diagnosable. The design approach ensures that a local diagnosis will be globally correct, because exactly one robot isolates the true fault and knows no remote faults could have occurred. Since the local diagnosers achieve a global diagnosis, this avoids the need for a centralized diagnosis coordinator [40].

Our distributed diagnosis algorithm considers a finite set of abrupt, persistent faults, and applies to single fault diagnosis for continuous systems. We denote the complete set of system faults as F, and the complete set of measurements as M. For a system of n robots, associated with each robot R_i is a set of local faults F_i and a set of local measurements M_i, such that

$$F = \bigcup_{1 \le i \le n} F_i \text{ and } M = \bigcup_{1 \le i \le n} M_i.$$

In distributed diagnosis, our objective is to design n diagnosers $\mathcal{D}_i \triangleq \mathcal{D}_{F_i, M_i^+}$, one for each robot R_i, so that $\mathcal{D}_i$ can diagnose all faults in F_i using M_i^+, where $M_i^+ \triangleq M_i \cup M_{ci}$, and M_{ci} are additional measurements from other robots. The design goal is to find the minimum set $M_{ci} \subseteq M$ such that each fault $f \in F_i$ can be uniquely isolated within the fault set F using M_i^+. That is, unique isolation in a local diagnoser corresponds to unique isolation in the global diagnoser. If $M_{ci} = \varnothing$, then we say that the diagnoser $\mathcal{D}_i$ is independent of the other diagnosers. Otherwise, to obtain globally correct

Algorithm 15 Distributed Diagnoser Design

Input: local fault sets F_i, local measurement sets M_i, fault signatures, ordering sets, k subsystems
for subsystem $i \in 1, \ldots, k$ **do**
  identify set $F_i' \subseteq F_i$ such that $f \in F_i'$ cannot be completely distinguished using M_i
  for $f \in F_i'$ **do**
    identify minimum set of communicated measurements to globally diagnose f
    add this set to the local measurement set

diagnoses for the faults in F_i, $\mathcal{D}_i$ must use additional measurements, so it is not independent. We will show in later that some robots' diagnosers will be independent.

The measurement set M_i^+ allows $\mathcal{D}_i$ to distinguish uniquely every fault $f \in F_i$ from the fault set F_i (local faults), and from $F - F_i$ (remote faults). Our design ensures that the effects observed on M_i^+ can only be explained by a single fault $f_{local} \in F_i$ or some (unknown) fault $f_{remote} \in F - F_i$ [28, 40]. Each robot R_i knows only the effects of faults in F_i on measurements in M_i^+. Therefore, if there is no $f \in F_i$ which matches the observations, the fault is guaranteed to be remote. Under the single fault assumption, agreement between individual diagnosers is reached implicitly. If $f \in F$ occurs, it will belong to exactly one F_i thus exactly one robot, R_i, will achieve the diagnosis $\{f\}$ and all other robots will eventually achieve the (empty) diagnosis $\varnothing$. Therefore, the global diagnosis is simply $\{f\}$. Practically, we do not have to wait for all other robots to complete their diagnostic tasks. A robot R_i may conclude that its diagnosis $\{f\}$ is the global diagnosis if one of three conditions holds: (i) all measurements in M_i^+ have deviated, so by design no other fault could have occurred, (ii) all other robots have reached the diagnosis $\varnothing$, thus leaving only $\{f\}$, or (iii) measurement deviation information allows the robot to conclude a remote fault could not have occurred.

The distributed diagnoser design algorithm generates the distributed diagnoser by minimizing the number of shared measurements between subsystems. For each subsystem, if a fault is not globally diagnosable using local measurements, it searches neighboring subsystems for a minimal set of additional measurements to make the fault globally diagnosable. The pseudocode is given as Algorithm 15, and extends the algorithm in [40, 41] using orderings. In the worst case, all combinations of measurements are considered, so the algorithm is exponential. From a practical viewpoint, since the diagnosers are built offline, their design time complexity is not of much concern.

Given each F_i, the distributed design algorithm determines the corresponding M_i^+ that each local diagnoser needs to have. Therefore, for robot R_i, we will obtain $\mathcal{D}_{F_i, M_i^+}$. To do this, we obtain the fault models for each $f \in F_i$ using M_i^+, i.e., $\mathcal{L}_{f, M_i^+}$. From these, we obtain each $\mathcal{D}_{\{f\}, M_i^+}$ and then compose these using the Δ composition described in Chapter VII.

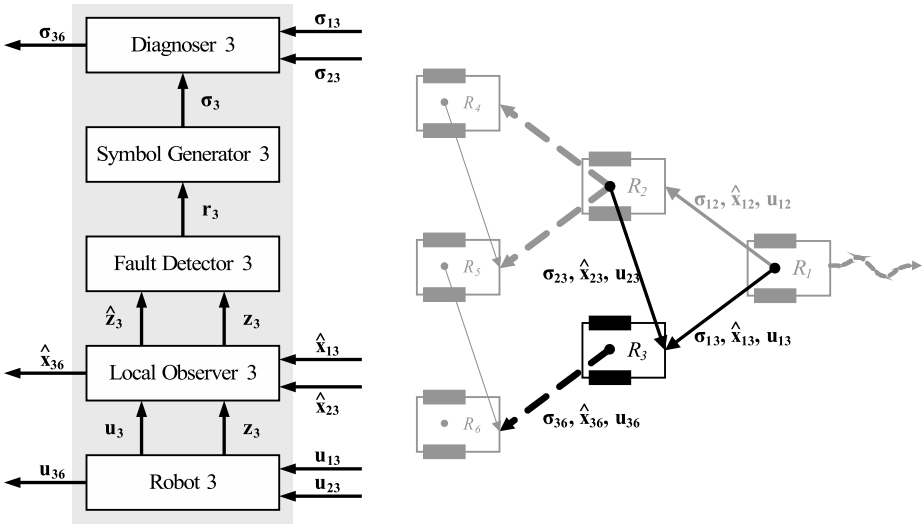


Figure 64: Diagnosis architecture for R_3

Distributed Diagnosis Architecture

The diagnosis architecture for the multi-robot system consists of distributed version of the same components as in the centralized case (Fig. 10). Fig. 64 illustrates the architecture for R_3 in the given six-robot formation. A robot R_i (e.g., R_3) receives communicated inputs $\mathbf{u_{ji}}$ (u_{13} and u_{23}) from each local leader R_j (R_1 and R_2). The local observers compute the output estimates, $\hat{\mathbf{z}}_\mathbf{i}$, ($\hat{\mathbf{z}}_\mathbf{3}$) given the input $\mathbf{u_i}$ ($\mathbf{u_3}$), the local measurements $\mathbf{z_i}$ ($\mathbf{z_3}$), and communicated state information, $\hat{\mathbf{x}}_\mathbf{ji}$ ($\hat{\mathbf{x}}_\mathbf{13}$ and $\hat{\mathbf{x}}_\mathbf{23}$), from each leader R_j (R_1 and R_2) as necessary. It also outputs relevant state information, $\hat{\mathbf{x}}_\mathbf{ik}$ ($\hat{\mathbf{x}}_\mathbf{36}$) to each follower R_k (R_6). The fault detectors for each robot compute the residuals of the measurements as differences between actual and predicted values, $\mathbf{r_i}$ ($\mathbf{r_3}$). If a fault is detected, the symbol generator computes qualitative values as events, $\sigma_\mathbf{i}$ ($\sigma_\mathbf{3}$), i.e., fault signatures, for the changes in measurement values. Each local diagnoser uses these signatures and communicated events $\sigma_\mathbf{ji}$ ($\sigma_\mathbf{13}$ and $\sigma_\mathbf{23}$) from each required robot R_j (R_1 and R_2) to isolate the fault. The local diagnoser also outputs some of its own signatures, $\sigma_\mathbf{ik}$ ($\sigma_\mathbf{36}$), required by the diagnoser design for other diagnosers $\mathcal{D}_k$ ($\mathcal{D}_6$) to use. We assume the communication delay is within one sample period.

We extend our traditional observer scheme to multi-robot systems by using a distributed, decentralized, extended Kalman filter (DDEKF) [154]. This method creates local observers for each robot, which share relevant observations and estimates. Each DDEKF produces estimates of the local state vector using local observations, local estimates, and required shared observations and estimates.

The DDEKF estimate update equations are given by:

$$\begin{aligned}
\hat{\mathbf{z}}_i(k) &= \mathbf{C}_i(k)\hat{\mathbf{x}}_i(k|k) \\
\hat{\mathbf{x}}_i(k|k) &= \mathbf{P}_i(k|k)\{\mathbf{P}_i^+(k|k-1)\hat{\mathbf{x}}_i(k|k-1) + \\
&\quad \sum_{j=1}^{n} \mathbf{P}_i^+(k|\mathbf{z}_j(k))\hat{\mathbf{x}}_i(k|\mathbf{z}_j(k))]\} \\
\mathbf{P}_i(k|k) &= [\mathbf{P}_i^+(k|k-1) + \sum_{j=1}^{n} \mathbf{P}_i^+(k|\mathbf{z}_j(k))]^+,
\end{aligned}$$

where $\mathbf{C}_i$ is the local output matrix, $\mathbf{P}_i$ the local covariance matrix, n the number of subsystems, and $^+$ indicates the generalized matrix inverse.

Distributed Diagnosers for Robot Formations

The formation system is defined by the global bond graph model. The set of faults for R_i is the set $F_i = \{MSe^-_{L,i}, MSe^-_{R,i}, G^-_{EL,i}, G^-_{ER,i}, B^+_{gyro,i}, B^-_{gyro,i}\}$, and the set of measurements is $M_i = \{v_{L,i}, v_{R,i}$. The complete fault set for the n robots is $F = \cup_i = 1^n F_i$ and the complete measurement set is $M = \cup_i = 1^n M_i$.

The TCG of the entire system is derived systematically from the global system bond graph model. It consists of a TCG for each robot, with additional edges between the robot TCG models that convey the measurements required by the formation control. These additional edges start at the local or remote measurement vertex and end at the effort source vertices representing actuator torque. For example, an edge is required from $f_{36,2}$ of R_2's TCG (v_2) to $e_{1,3}$ of R_3's TCG (τ_{L3}) because R_3's control requires v_2 as an input. This represents the fact that the torque τ_{L3} is causally influenced by v_2. The labels on these edges include a dt specifier to indicate a time delay due to the system dynamics. The sign of the label depends on whether a change in the measurement will cause a direct or inverse change on the control output. Because the control is nonlinear, the direction of change will depend on the robot's position. The effects of x, y, and θ depend on position, but the control output change due to a fault transient will always initially change in the same direction as the communicated velocity measurements. These edges capture the qualitative effects of the measurements in the transient dynamics of the robot's motion. Therefore, the global system TCG not only captures fault propagation within a single robot but also from one robot to another through the leader-follower interactions.

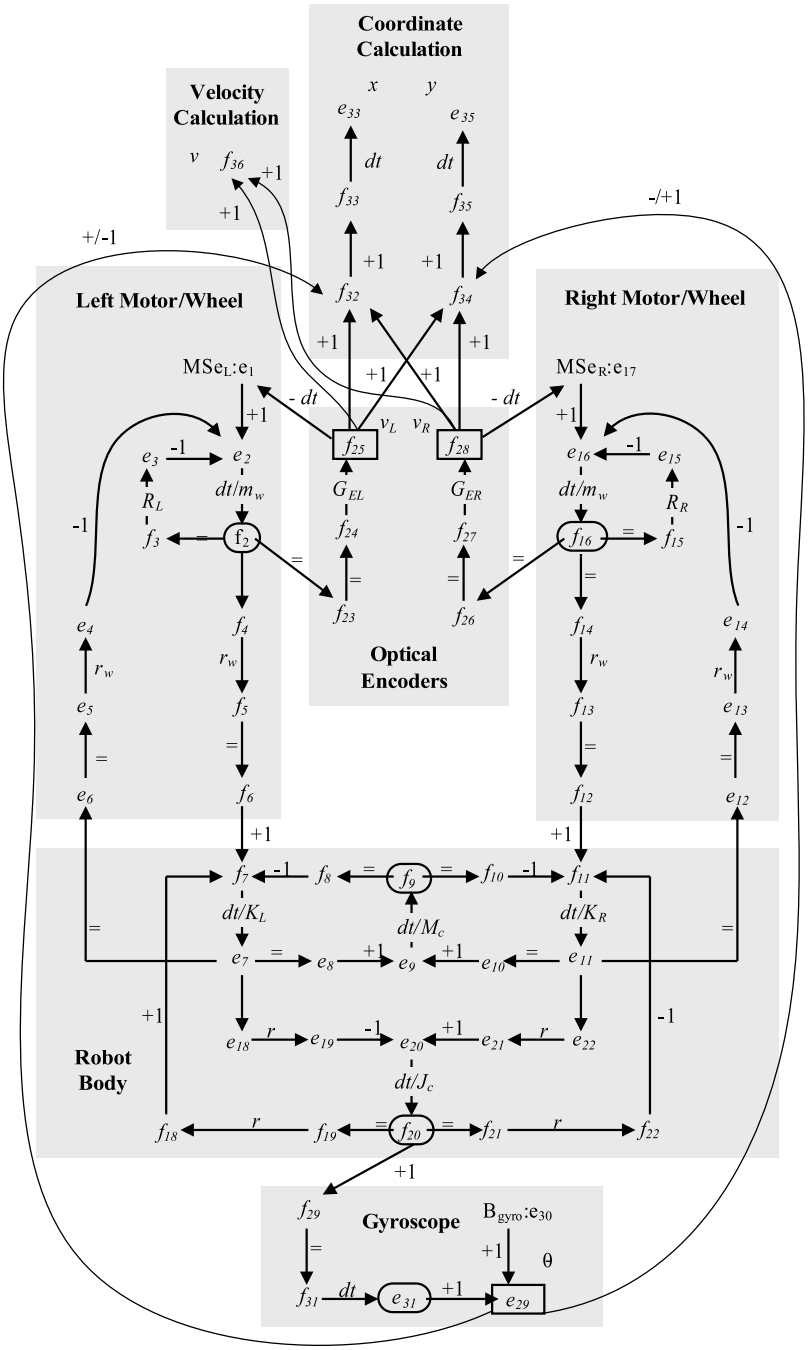


Figure 65: TCG for a single robot of the multi-robot system.

Fault Propagation Graph

The effects of a fault in a single robot may propagate to other robots in the system through the control links, and we take this into account in constructing the global TCG. For example, in Fig. 63, if an actuator fault occurs in R_2, it cannot maintain its separation and bearing with respect to R_1. Because R_1 will continue on its predefined trajectory, R_3 will lose its ability to maintain its

pre-specified separation to both R_1 and R_2. Therefore, the fault in R_2 has now propagated to R_3. Since R_3 can no longer act as a leader to its followers, the formation will not be maintained, and the fault will further propagate to R_6. On the other hand, R_4 can still maintain its separation and bearing with respect to R_2, and thus R_5 can also maintain its separations to R_2 and R_4. Depending on the situation and control strategy employed, faults will, therefore, propagate to some parts of the system and not others. To make the overall diagnosis process more efficient, we can remove causal links between the robots where faults do not propagate, and still generate correct diagnosis results. The reduced interactions between the robots are captured in the form of a *fault propagation graph* that is derived from the control graph.

Definition 55 (Fault Propagation Graph)**.** A fault propagation graph $\mathcal{G}$ is a directed, acyclic graph, where each robot, R_i, defines a vertex of the graph. A directed edge (R_i, R_j) implies that faults may propagate from R_i to R_j. We denote the parents of a robot R_i in $\mathcal{G}$ as $Par(R_i)$ and the ancestors as $Anc(R_i)$.

Fig. 66 shows the fault propagation graph for the six-robot formation. To improve the efficiency of diagnosis, the fault propagation graph is constructed as a subset of the formation control graph. An edge (R_i, R_j) in a formation control graph $\mathcal{C}$ is not included in the corresponding graph $\mathcal{G}$, if R_j has only one incoming edge. Faults do not propagate to robots with single incoming edges, i.e., robots that have a single leader, because that robot can maintain its position relative to its leader for any arbitrary trajectory if it is not faulty itself. In the example formation illustrated in Fig. 63, edges (R_1, R_2) and (R_2, R_4) are removed because R_2 and R_4 have single leaders. An edge (R_i, R_j) from the control graph $\mathcal{C}$ is also removed if R_j has another edge (R_k, R_j) and R_k has a single incoming edge (R_i, R_k). Faults also do not propagate along these edges, because a fault in R_i would not propagate to R_k, and since R_j depends on both R_i and R_k, it will not exhibit faulty behavior if R_i does. In the example, edges (R_1, R_3) and (R_2, R_5) are removed. A simple algorithm can be constructed to find and remove all such edges. Note that only robots employing SBC can become source vertices, i.e., have no incoming edges, through this procedure.

The fault propagation graph describes whether to treat control information as inputs (through which faults do not propagate) or remote measurements (through which faults do propagate). The fault detection model can, therefore, be simplified with respect to $\mathcal{G}$. An absent link from R_i and R_j in $\mathcal{G}$ indicates that R_j does not require estimates from R_i to produce its local estimates.

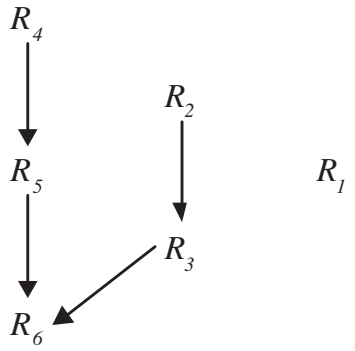


Figure 66: Fault propagation graph for the formation of six robots.

Diagnosability Analysis

An important prerequisite for distributed diagnoser design is to determine whether the system is diagnosable, i.e., all faults of interest can be uniquely isolated with the given measurement set. A fault f_1 will be distinguished from another fault f_2 if, during the isolation process, a measurement deviation occurs that matches the fault signature for f_1 but not f_2.

Table 8 shows the fault signatures for faults and measurements of two robots, R_2 and R_3, in the six-robot formation. The signatures are generated from the system TCG, with only the magnitude change (discontinuity) symbol and the first non-zero direction of change symbol shown. A `*` symbol indicates an indeterminate effect, i.e., there are at least two paths of the same order that propagate `+` and `-` effects, and, therefore, the sign of the change cannot be computed using qualitative propagation. Some of the effects of these faults are determined by the robot's position, since the controller inputs are functions of separations and bearings, which are functions of position. Such effects are denoted by a `0*`. A `00` indicates that a fault has no effect on the corresponding measurement because there is no path to it.

From the signatures, it is clear that not all faults can be distinguished. If some fault f occurs in R_2, one of R_2's measurements will deviate. Because, faults in R_3 do not propagate to R_2's measurements, the fault f cannot belong to R_3. This is indicated by the absence of a causal path from R_3 to R_2 and qualified by the `00` symbols in the lower left segment of Table 8. However, if an actuator fault occurs in R_3, deviations could match those of one of R_2's faults. Since R_2's measurements will never deviate if one of these faults occurs, we would have to wait infinitely long before we can be certain the fault does not belong to R_2. For example, if MSe^-_{L3} occurs, its effects on R_3's measurements could manifest as `0-` on v_{L3}, `0-` on v_{R3}, `0-` on v_3, and `0+` on θ_3, which can be explained by any of the faults for R_2. In general, we cannot distinguish between actuator faults

Table 8: Fault Signatures for R_2 and R_3 in the Six-robot Formation

Fault	v_{L2}	v_{R2}	v_2	θ_2	v_{L3}	v_{R3}	v_3	θ_3
$MSe^-_{L,2}$	0−	0∗	0−	0+	0∗	0∗	0−	0∗
$MSe^-_{R,2}$	0∗	0−	0−	0−	0∗	0∗	0−	0∗
$G^-_{EL,2}$	−+	0∗	−+	0−	0∗	0∗	0−	0∗
$G^-_{ER,2}$	0∗	−+	−+	0+	0∗	0∗	0−	0∗
$B^+_{gyro,2}$	0+	0−	0∗	+−	0∗	0∗	0∗	0∗
$B^-_{gyro,2}$	0−	0+	0∗	−+	0∗	0∗	0∗	0∗
$MSe^-_{L,3}$	00	00	00	00	0−	0∗	0−	0+
$MSe^-_{R,3}$	00	00	00	00	0∗	0−	0−	0−
$G^-_{EL,3}$	00	00	00	00	−+	0∗	−+	0−
$G^-_{ER,3}$	00	00	00	00	0∗	−+	−+	0+
$B^+_{gyro,3}$	00	00	00	00	0+	0−	0∗	+−
$B^-_{gyro,3}$	00	00	00	00	0−	0+	0∗	−+

occurring on different robots in the formation. Therefore, using the given measurement set and the fault signature approach, the system is not globally diagnosable. This motivates the need for employing additional discriminatory information to achieve global diagnosability.

Using relative measurement orderings in addition to fault signatures, actuator faults can be globally distinguished. From the global TCG model, it follows that an actuator fault will appear first in the velocity measurement of that wheel and then in other measurements. Taking the example from Table 8, if $MSe^-_{L,3}$ occurs, it will manifest first in v_{L3}. Since none of R_2's measurements have yet deviated, we know that the fault is local to R_3. Using relative measurement orderings, actuator faults occurring on different robots can be distinguished.

Table 9 shows the relative measurement orderings for the fault parameters of R_2, for measurements associated with R_2, R_3, and R_6 in the six-robot formation. Orderings implied by transitivity are omitted from the table. As evidenced by Table 9, faults manifest first in their associated measurement before other measurements in the system (e.g., actuator and encoder faults manifest first in velocity measurements of that wheel).

Because faults cannot propagate in the opposite direction, faults in R_3 and R_6 will both have orderings in the format of $m_i \prec_f m_j$, where m_i is a local measurement, f is a local fault, and m_j is a measurement of R_2. If a fault occurs in R_2, either v_2 or θ_2 will deviate before any measurement in R_3. Therefore, to distinguish between R_2's faults and R_3's faults, one of these measurements will be useful. Which one is useful depends on whether the fault is an actuator or encoder fault (where v_2 is useful) or a gyroscope fault (where θ_2 is useful). This results in the following lemma.

Lemma 4. *Faults appearing in a parent $R_p \in Par(R_i)$ in $\mathcal{G}$ can be distinguished from faults appearing in R_i using orderings for both v_p and θ_p and local measurements of R_i.*

Table 9: Relative Measurement Orderings for R_2 in the Six-robot Formation for Measurements of R_2, R_3, and R_6

Faults	Relative Measurement Orderings
$MSe_{L,2}, G_{EL,2}$	$v_{L2} \prec v_{R2}, v_{L2} \prec \theta_2, v_2 \prec v_{R2}, v_2 \prec \theta_2,$ $v_{L2} \prec v_{L3}, v_{L2} \prec v_{R3}, v_{L2} \prec v_3, v_2 \prec v_{L3}, v_2 \prec v_{R3}, v_2 \prec v_3,$ $v_3 \prec \theta_3, v_3 \prec v_6,$ $v_{L2} \prec v_{L6}, v_{L2} \prec v_{R6}, v_{L2} \prec v_6,$ $v_2 \prec v_{L6}, v_2 \prec v_{R6}, v_2 \prec v_6,$ $v_6 \prec \theta_6$
$MSe_{R,2}, G_{ER,2}$	$v_{R2} \prec v_{L2}, v_{R2} \prec \theta_2, v_2 \prec v_{R2}, v_2 \prec \theta_2,$ $v_{R2} \prec v_{L3}, v_{R2} \prec v_{R3}, v_{R2} \prec v_3, v_2 \prec v_{L3}, v_2 \prec v_{R3}, v_2 \prec v_3,$ $v_3 \prec \theta_3, v_3 \prec v_6,$ $v_{R2} \prec v_{L6}, v_{R2} \prec v_{R6}, v_{R2} \prec v_6,$ $v_2 \prec v_{L6}, v_2 \prec v_{R6}, v_2 \prec v_6,$ $v_6 \prec \theta_6$
$B^+_{gyro,2}, B^-_{gyro,2}$	$\theta_2 \prec v_{L2}, \theta_2 \prec v_{R2}, \theta_2 \prec v_2$ $\theta_2 \prec v_{L3}, \theta_2 \prec v_{R3}, \theta_2 \prec v_3,$ $v_3 \prec \theta_3, v_3 \prec v_6,$ $\theta_2 \prec v_{L6}, \theta_2 \prec v_{R6}, \theta_2 \prec v_6,$ $v_6 \prec \theta_6$

Proof. Given $R_p \in Par(R_i)$, all f_i of R_i do not manifest in R_p, because there is no causal path from R_i to R_p since $\mathcal{G}$ is acyclic. So, we have the orderings $m_i \prec_{f_i} v_p$ and $m_i \prec_{f_i} \theta_p$ for each f_i of R_i. From the TCG analysis, each fault f_p of R_p passes through either v_p or θ_p before any measurement of R_i, thus resulting in either $v_p \prec_{f_p} m_i$ or $\theta_p \prec_{f_p} m_i$ for all m_i of R_i (Theorem 1). Therefore, the ordering sets will always conflict and we can use this information to distinguish among the faults. □

It is important to note that we cannot derive orderings comparing a left or right velocity of R_3 to a left or right velocity of R_6. This is because the path to a left or right velocity measurement of R_6 could go through either the left or right velocity of R_3, and we don't know which path is faster. Thus, if an actuator fault occurs in R_2, R_3's left and right velocity measurements are useless to distinguish between local and remote actuator faults. However, we do have the ordering $v_3 \prec v_6$ for all of R_2's faults. Thus we can use v_3 to distinguish between actuator faults in R_1 and R_6. The ordering $v_3 \prec v_6$ essentially says that either v_{L3} or v_{R3} will deviate before any measurements of R_6, since remote faults affect the velocity measurement first ($v_6 \prec \theta_6$ for all remote faults). It does not matter which of v_{L3} or v_{R3} deviates first, only knowing that one will deviate is helpful. So although v_2 provides no extra discriminatory information in terms of fault signatures, it is helpful in terms of measurement orderings. This results in the following lemma.

Lemma 5. *Faults appearing in an ancestor $R_a \in Anc(R_i)$ in $\mathcal{G}$ such that $R_a \in Anc(R_p)$ and $R_p \in Par(R_i)$, can be distinguished from faults appearing in R_i using orderings for v_p and local measurements of R_i.*

Proof. Given $R_p \in Par(R_i)$, and $R_a \in Anc(R_p)$, all f_i of R_i do not manifest in R_p, because there is no causal path from R_i to R_p since $\mathcal{G}$ is acyclic. So, we have the ordering $m_i \prec_{f_i} v_p$ for each f_i of R_i. From the TCG analysis, each fault f_a of R_a passes through v_p before any measurement of R_i, thus resulting in $v_p \prec_{f_a} m_i$ for all m_i of R_i (Theorem 1). Therefore the ordering sets will always conflict, and we can distinguish the faults. □

The following theorem shows how local and remote faults can be discriminated.

Theorem 8. *A fault is local if and only if a local measurement deviates before a remote measurement.*

Proof. If a fault is local, a local measurement will deviate before any remote measurement because for every local fault there is some set of local measurements that deviate before every other measurement. If a local measurement deviates before a remote measurement, the fault must be local because for all non-local faults, the fault will manifest in a parent (if the fault originated in an ancestor) before the local robot (Lemmas 4 and 5), or in a child before the local robot (because faults in a child never manifest in their parents). □

Distributed Diagnoser Design

For the formation system, the subsystems are the individual robots. The diagnoser for R_i is responsible for diagnosing faults in the set $F_i = \{MSe^-_{L,i},\ MSe^-_{R,i},\ G^-_{EL,i},\ G^-_{ER,i},\ B^+_{gyro,i},\ B^-_{gyro,i}\}$ using measurements $M_i = \{v_{L,i},\ v_{R,i},\ v_i,\ \theta_i\}$, i.e., each robot is responsible for diagnosing faults in its components using its local measurements.

Running the algorithm shows that each robot must be communicated the velocity and heading measurements of its parents in the fault propagation graph. This ensures that each robot has enough information to produce an independent, globally correct diagnosis. From Lemma 4, each robot will need both v and θ measurements of each parent in the fault propagation graph, in order to distinguish between local faults and those appearing in the parents. From Lemma 5, these measurements are enough to distinguish between local faults and those appearing in the ancestors in the fault propagation graph, therefore, these are the minimal communicated measurements.

Table 10: Distributed Diagnoser Design for the Six-robot Formation

Robot	Fault Set	Measurement Set
R_1	$\{MSe^-_{L,1}, MSe^-_{R,1}, G^-_{EL,1}, G^-_{ER,1}, B^+_{gyro,1}, B^-_{gyro,1}\}$	$\{v_{L,1}, v_{R,1}, \theta_1\}$
R_2	$\{MSe^-_{L,2}, MSe^-_{R,2}, G^-_{EL,2}, G^-_{ER,2}, B^+_{gyro,2}, B^-_{gyro,2}\}$	$\{v_{L,2}, v_{R,2}, \theta_2\}$
R_3	$\{MSe^-_{L,3}, MSe^-_{R,3}, G^-_{EL,3}, G^-_{ER,3}, B^+_{gyro,3}, B^-_{gyro,3}\}$	$\{v_{L,3}, v_{R,3}, \theta_3, v_2, \theta_2\}$
R_4	$\{MSe^-_{L,4}, MSe^-_{R,4}, G^-_{EL,4}, G^-_{ER,4}, B^+_{gyro,4}, B^-_{gyro,4}\}$	$\{v_{L,4}, v_{R,4}, \theta_4\}$
R_5	$\{MSe^-_{L,5}, MSe^-_{R,5}, G^-_{EL,5}, G^-_{ER,5}, B^+_{gyro,5}, B^-_{gyro,5}\}$	$\{v_{L,5}, v_{R,5}, \theta_5, v_4, \theta_4\}$
R_6	$\{MSe^-_{L,6}, MSe^-_{R,6}, G^-_{EL,6}, G^-_{ER,6}, B^+_{gyro,6}, B^-_{gyro,6}\}$	$\{v_{L,6}, v_{R,6}, \theta_6, v_3, \theta_3, v_5, \theta_5\}$

Additionally, the local v measurement is not necessary to distinguish local and remote faults because v_L and v_R provide the same information in this respect[1]. Essentially, the discriminatory information that the remote v and θ measurements provide is that if a remote fault occurs, it will manifest in one of the remote v or θ measurements before any local measurement, thus allowing the diagnosers to distinguish between local and remote faults.

Table 10 illustrates the individual fault and measurement sets for the diagnosers in the six-robot formation. It is important to note that not all robots require remote measurements to determine a globally correct local diagnosis. Some of the robots (R_1, R_2, and R_4) require only local measurements, i.e., these diagnosers are independent. This is the case when the robot is a source vertex in the fault propagation graph, and occurs because the fault effects of other robots cannot propagate to it. Therefore, any fault effects it observes are known to be caused by a local fault.

The pruned distributed diagnoser for R_i is given in Fig. 67. When a fault trace is observed that it does not accept, an empty diagnosis is returned. For example, in the six-robot formation, R_3 requires v_2 and θ_2. If a fault in R_2 occurs, then v_2 or θ_2 will be observed as the first deviation. But, $\mathcal{D}_3$ does not accept either of those as the first event, therefore determining that the fault is remote.

These results easily extend to arbitrary formations that satisfy the constraints of [139, 140]. The fault propagation graph $\mathcal{G}$ can be derived from the control graph $\mathcal{C}$. The diagnoser design is direct from $\mathcal{G}$ since each robot needs the velocity and heading measurements of each parent in $\mathcal{G}$.

[1] Alternatively, v could be kept and v_L and v_R dropped, because the system would still be diagnosable. However, we opt to keep v_L and v_R instead so that we do not have to wait for θ to deviate in order to distinguish between faults of the left and right wheels.

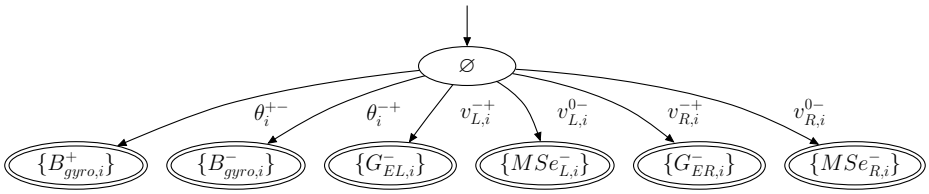


Figure 67: Pruned diagnoser for R_i

Scalability

The scalability of the approach can be characterized using two metrics, the size of the diagnoser and the number of communicated measurements, quantifying the computational and communication requirements, respectively. Let F represent the complete fault set, M the complete measurement set, and n the number of robots. Also, let F_i be the fault set of R_i, and M_i^+ be the measurement set for R_i determined by the distributed diagnoser design algorithm. In a centralized approach, the central diagnoser must diagnose all faults in F using measurements in M. The size of the diagnoser is then $S_C = |F||M| + |F||M|^2$ so that it can store both signatures and orderings for each fault. In the replicated centralized diagnoser approach, each diagnoser is of size S_C, resulting in nS_C space. In the proposed distributed approach, however, R_i must only diagnose faults in F_i using measurements in M_i^+, resulting in space complexity $S_i = |F_i||M_i| + |F_i||M_i^+|^2$. The total space required for all individual distributed diagnosers will always be less than that of a centralized diagnoser, i.e., $\sum_i S_i < S_C$ if not all measurements are communicated. The reason is that some of the information is discarded because it is not useful in the local diagnosers. For example, we don't need to store anywhere the effects of R_2's faults on R_1's measurements, because none of R_1's measurements are needed to diagnose R_2's faults. Diagnoser size directly relates to diagnostic efficiency. The smaller the diagnoser size, the smaller the number of faults and measurements to consider, and thus the greater its computational efficiency.

The number of communicated measurements characterize the communication overhead incurred by the distributed algorithm. In a centralized approach, each robot must communicate its measurements to the centralized diagnoser, resulting in a total of $|M|$ communicated measurements. In a replicated centralized diagnoser approach, each robot would have to communicate its measurements to all other robots, resulting in a total of $\sum_i (n-1)|M_i|$ communicated measurements. In our distributed approach, however, communication is minimized by the diagnoser design algorithm. From Lemmas 4 and 5, only the velocity (v) and heading (θ) measurements are required from each parent in the fault propagation graph. Therefore, at most two measurements must be communicated to each robot (except the formation leader) from each local leader (at most two), resulting in at most

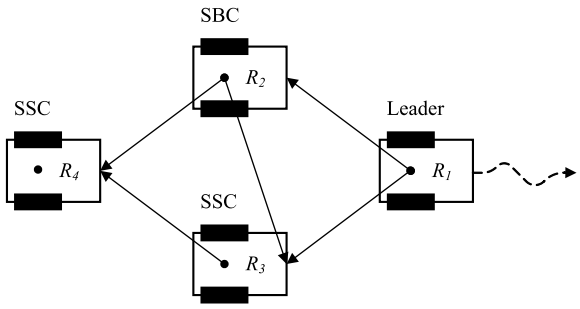


Figure 68: Experimental setup.

$4(n-1)$ communicated measurements for the worst case. Hence, the number of communicated measurements required per robot is independent of formation size. The total number of communicated measurements for all robots is linear in the formation size, so, like the centralized case, the approach scales linearly with large formations. In the six-robot example used throughout the paper, there are 4 edges in $\mathcal{G}$, resulting in a total of 8 communicated measurements for the distributed approach. For a centralized approach, since each robot has 3 measurements in its measurement set, 18 measurements must be communicated.

Experimental Results

The effectiveness of the distributed detection and isolation algorithms is demonstrated in a laboratory setting with four ActivMedia Pioneer 3-DX mobile robots moving in the formation illustrated in Fig. 68. For the DDEKF, each follower must know the position estimates of x, y, θ, and v for each local leader. Each robot observes its own wheel velocities and heading, i.e., the local measurements are $\mathbf{z}_i = [v_{Li}\ v_{Ri}\ \theta_i]^T$ for R_i. The robots communicate over an 802.11b wireless ad-hoc network. Fig. 69 shows the nominal trajectories of the robots moving at a pre-specified speed of 0.1 m/s. The experiment is run for 40 s. The top plot shows the robot trajectories, with their starting and ending locations drawn. The lower left plot shows the robot velocities, and the lower right plot shows their headings. All the robots maintain the shape variables, so the formation is maintained. All faults listed in Table 7 were introduced through software. The sampling period of the distributed controllers and diagnosers was 0.1 s. At the selected sampling rate, the packet loss was negligible (measured less than 0.1%). Since communication is expected at the selected sampling rate, persistent errors in the network links can be easily diagnosed by software (viewed as additional diagnosers) and are not considered here.

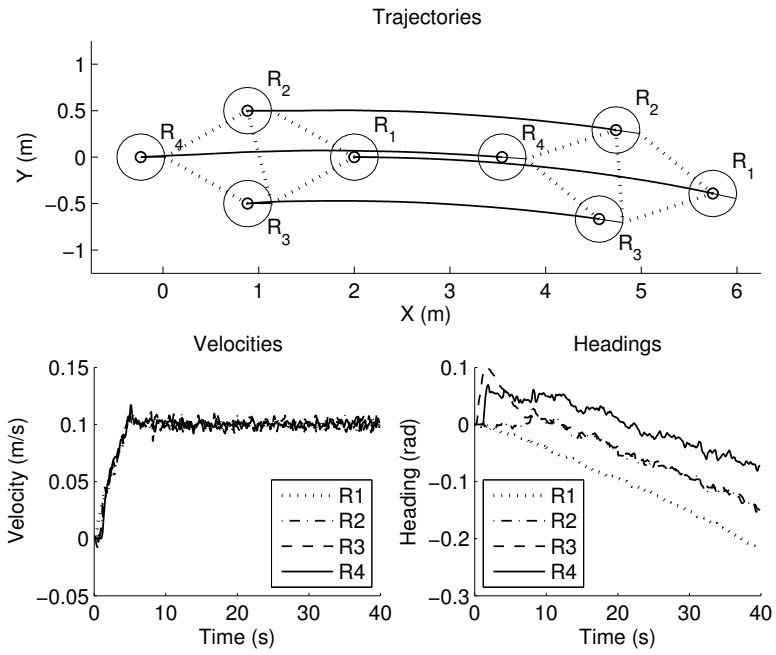


Figure 69: Nominal system behavior.

For this four-robot formation, the fault propagation graph includes only the edges (R_2, R_3), (R_2, R_4), and (R_3, R_4). Therefore, R_3 requires measurements from R_2 (v_2 and θ_2), and R_4 requires measurements from both R_2 and R_3 (v_2, θ_2, v_3, and θ_3). R_1 and R_2 are source vertices so they require only local measurements.

In the following, we illustrate our approach for an actuator fault of the left wheel of R_2 ($MSe^-_{R,2}$) at a magnitude of 0.05 m/s, i.e., the wheel velocity saturates at half the desired speed. Fig. 70 shows the faulty trajectories for the robots, and Table 11 traces the diagnosis steps. Initially, the diagnosers assume empty fault sets. The fault is injected at $t = 20.0$ s. It causes the left wheel to slow down, therefore, the heading deviates, and the right wheel begins to speed up to keep its separation with R_1. R_3 and R_4 begin to slow down to maintain their positions with respect to R_2. A deviation in $v_{L,2}$ at 20.4 s triggers R_2's fault isolation procedure. Six steps later the deviation is determined not to be discontinuous, i.e., the change in the measurement is smooth, not abrupt.

R_2 starts with its entire fault set, F_2, as the set of possible candidates. As predicted, $v_{L,2}$ is the first deviation, so based on orderings, the fault set is reduced to the faults of the left wheel. The change of $v_{L,2}$ matches the fault signature of `0-`, thus isolating the fault to be $MSe^-_{L,2}$. By design, this is guaranteed to be the globally unique fault, so recovery actions may commence and the other robots notified. Only one measurement deviation was needed to obtain a global diagnosis, so this

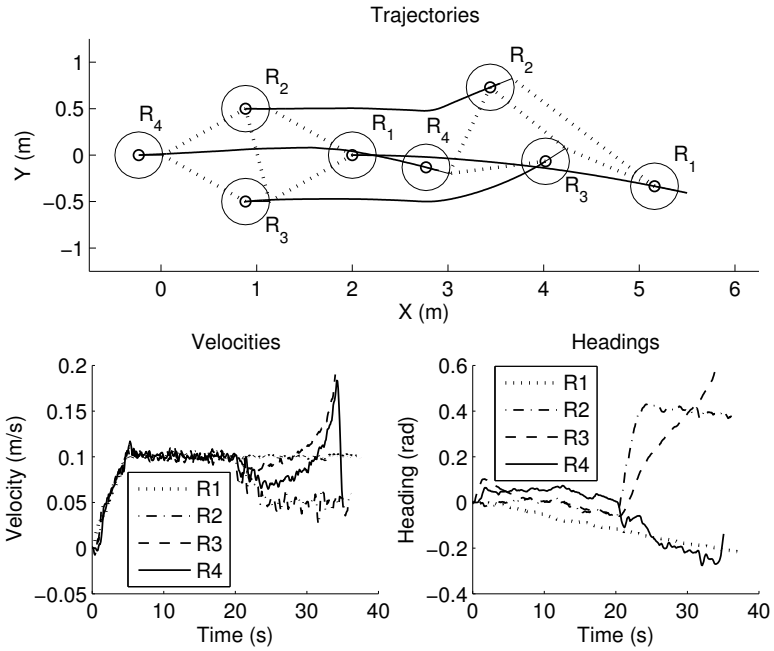


Figure 70: System behavior with $MSe^{-}_{R,2}$ occurring with 0.05 m/s magnitude.

Table 11: Diagnosis Trace for Left Actuator Fault of R_2

Time	Event	R_1	R_2	R_3	R_4
20.0	Fault injected (R_2)	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
20.4	Fault detected (R_2)	$\emptyset$	F_2	F_3	F_4
	v_{L2} deviates	$\emptyset$	$\{MSe^{-}_{L,2}, G^{-}_{EL,2}\}$	$\emptyset$	$\emptyset$
20.7	θ_2 deviates	$\emptyset$	$\{MSe^{-}_{L,2}, G^{-}_{EL,2}\}$	$\emptyset$	$\emptyset$
21.0	v_{L2} 0- determined	$\emptyset$	$\{MSe^{-}_{L,2}\}$	$\emptyset$	$\emptyset$
	Diagnosis (R_2)	$\emptyset$	$MSe^{-}_{L,2}$	$\emptyset$	$\emptyset$

demonstrates the efficiency of using relative measurement orderings in fault isolation. Because a communicated remote measurement (v_2) has deviated before any local measurements, R_3 and R_4 can eliminate all their local faults and determine a remote fault has occurred. R_1 does not observe a deviation in any of its local measurements so it does not produce a diagnosis.

We illustrate our approach now for an encoder fault of the right wheel of R_2 ($G^{-}_{ER,2}$) at a magnitude of 30%, i.e., the encoder misses 30% of its counts. Fig. 71 shows the faulty trajectories for the robots, and Table 12 traces the diagnosis steps. Initially, the diagnosers assume empty fault sets. The fault is injected at $t = 20.0$ s. It causes an abrupt decrease in the right velocity measurement, causing the right wheel to speed up. Therefore, the heading deviates, and the left wheel begins to slow down to keep its separation with R_1. R_3 and R_4 begin to speed up to maintain

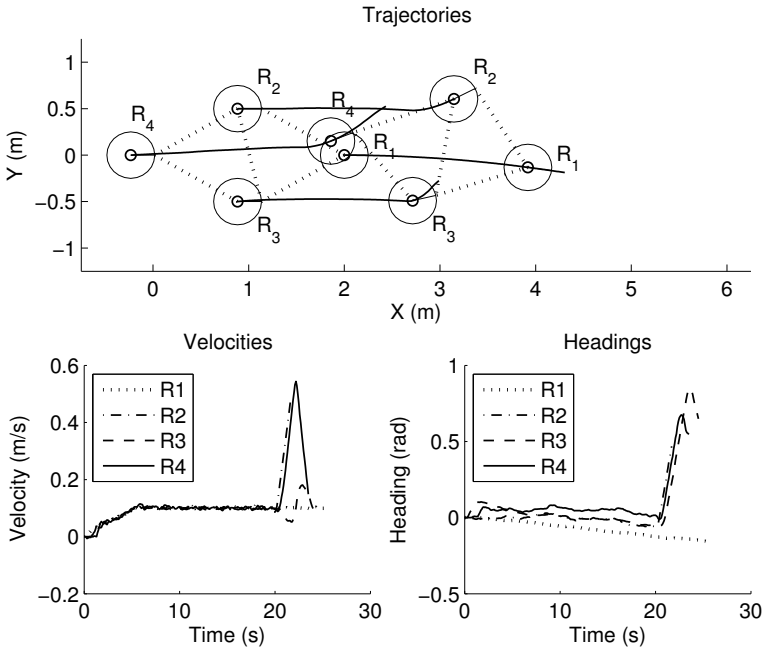


Figure 71: System behavior with $G^{-}_{ER,2}$ occurring with 30% magnitude.

their positions with respect to R_2. A deviation in $v_{R,2}$ at 20.1 s triggers R_2's fault isolation procedure. Six steps later the deviation is labelled as discontinuous.

R_2 starts with its entire fault set, F_2, as its set of possible candidates. As predicted, $v_{R,2}$ is the first deviation, so based on orderings, the fault set is reduced to only faults of the right wheel. The change of $v_{R,2}$ matches the fault signature of `-+`, thus isolating the fault to be $G^{-}_{ER,2}$. Again, by design, this is guaranteed to be the globally unique fault. For this example too, only one measurement deviation was needed to obtain a global diagnosis, so this demonstrates the efficiency of using relative measurement orderings in fault isolation. Because a communicated remote measurement (v_2) has deviated before any local measurements, R_3 and R_4 can eliminate all their local faults and determine a remote fault has occurred. R_1 does not observe a deviation in any of its local measurements so it does not produce a diagnosis.

All faults of interest, listed in Table 7, were successfully isolated using the distributed diagnoser. The summary of the diagnosis results is shown in Table 13. Due to the high discriminatory power the combination of fault signatures and relative measurement orderings provide, all faults could be isolated with only a single measurement deviating. The magnitude of the fault and its time of injection are shown, along with all measurement deviations observed until a global diagnosis is known. All faults were injected at 20 s. Beside each measurement deviation is the time of detection

Table 12: Diagnosis Trace for Right Encoder Fault of R_2

Time	Event	R_1	R_2	R_3	R_4
20.0	Fault injected (R_2)	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
20.1	Fault detected (R_2)	$\emptyset$	F_2	F_3	F_4
	v_{R2} deviates	$\emptyset$	$\{MSe^-_{R,2}, G^-_{ER,2}\}$	$\emptyset$	$\emptyset$
20.4	v_{R4} deviates	$\emptyset$	$\{MSe^-_{R,2}, G^-_{ER,2}\}$	$\emptyset$	$\emptyset$
20.5	v_{L2} deviates	$\emptyset$	$\{MSe^-_{R,2}, G^-_{ER,2}\}$	$\emptyset$	$\emptyset$
20.6	θ_2 deviates	$\emptyset$	$\{MSe^-_{R,2}, G^-_{ER,2}\}$	$\emptyset$	$\emptyset$
20.7	v_{L2} -+ determined	$\emptyset$	$\{G^-_{ER,2}\}$	$\emptyset$	$\emptyset$
	Diagnosis (R_2)	$\emptyset$	$G^-_{ER,2}$	$\emptyset$	$\emptyset$

Table 13: Diagnosis Results for the Four-robot Formation

Fault	Magnitude	Diagnosis Trace
$MSe^-_{L,2}$	90 mm/s	v_{L2} 0- 20.4-21.0 s
$MSe^-_{L,2}$	70 mm/s	v_{L2} 0- 20.3-20.9 s
$MSe^-_{L,2}$	50 mm/s	v_{L2} 0- 20.4-21.0 s
$G^-_{ER,2}$	10%	v_{R2} -+ 20.1-20.6 s
$G^-_{ER,2}$	30%	v_{R2} -+ 20.1-20.7 s
$G^-_{ER,2}$	50%	v_{R2} -+ 20.0-20.0 s
$B^+_{gyro,2}$	0.08 rad	θ_2 +- 20.1-20.1 s
$B^+_{gyro,2}$	0.1 rad	θ_2 +- 20.1-20.1 s
$B^-_{gyro,2}$	-0.08 rad	θ_2 -+ 20.1-20.1 s
$B^-_{gyro,2}$	-0.1 rad	θ_2 -+ 20.0-20.0 s
$MSe^-_{L,3}$	50 mm/s	v_{L3} 0- 20.4-21.0 s
$MSe^-_{R,3}$	50 mm/s	v_{R3} 0- 20.3-20.9 s

followed by the time at which it was determined whether or not a discontinuity occurred. The approach is applicable for smaller fault magnitudes, as long as the fault detector is appropriately tuned. If the fault detector and symbol generation work correctly, then by construction the fault isolation will always execute correctly. The fault detector was tuned for the laboratory setting and the fault magnitudes under consideration. Multiple experiments were performed to achieve reliable fault detection, and, therefore, no false positives occurred. Because the fault magnitudes were sufficiently large compared to the system noise, false negatives did not occur either.

Summary

In this case study we described an approach for distributed diagnosis in formations of mobile robots. We derived the system model encompassing the plant, sensors, actuators, communication, and control. The DDEKF scheme was applied for distributed estimation and tracking of nominal system behavior, and the Z-test was used for robust fault detection. The qualitative fault isolation scheme combined the use of fault signatures and relative measurement orderings, increasing the discrimina-

tory power of the measurement sets. Measurement orderings were shown to be necessary to ensure diagnosability in the formation systems studied in this paper. Using both signatures and orderings, diagnosers can require fewer measurements, and diagnosis results are achieved faster. Distributed diagnosers were designed from a global system model, and the diagnosis scheme was shown to scale well with formation size. The design was such that each local diagnosis was globally correct, thus circumventing the need for a centralized coordinator. Experimental results demonstrated the validity and usefulness of the approach.

In contrast to previous work in mobile robot diagnosis, our approach can be applied efficiently to diagnosis of process, sensor, and actuator faults in robot formations in a distributed fashion by employing relative measurement orderings. In addition to easily handling multiplicative faults, our approach qualifies residuals with a richer feature set than parity relations approaches and incorporates temporal information, resulting in increased discriminatory power of the measurements. Quantitative techniques, like particle filtering, do not scale well with the number of possible faults and are difficult to distribute among multiple robots with limited computational resources. We use qualitative fault isolation instead which is very efficient but currently is limited to abrupt faults. To deal with parametric uncertainty, we incorporate model uncertainty as a parameter in our fault detection scheme and apply a statistical test on the residuals to robustly detect faults. We use a single distributed Kalman filter as opposed to using a bank of Kalman filters, which requires a Kalman filter for each fault and is not efficient for distributed systems with a large number of faults. None of the previous approaches explicitly use any temporal measurement deviation information to resolve ambiguities in the diagnosis results. Relative measurement orderings distinguish among faults based on event orderings, where the events are measurement deviations. The technique, therefore, has some similarities to discrete-event diagnosis approaches in [5, 6, 62], and decentralized approaches in [155]. In contrast to other distributed diagnosis approaches, we formulate distributed diagnosis as a design problem, creating local diagnosers which are guaranteed to have enough information such that no coordination needs to occur, thus local diagnosis results correspond to global diagnosis results.

CHAPTER IX

CASE STUDY: ADVANCED DIAGNOSTICS AND PROGNOSTICS TESTBED

Fault diagnosis is crucial for ensuring the safe operation of complex engineering systems. Faults and degradations need to be quickly identified so that corrective actions can be taken, and catastrophic situations avoided. Most real-world, embedded systems are hybrid in nature. In such systems, a discrete abstraction may be adequate for model-based diagnosis [4]. Purely discrete techniques, however, are inadequate for systems that combine complex continuous and discrete behaviors [16,97]. Hybrid models have to be employed for correct tracking and diagnosis.

The Advanced Diagnostics and Prognostics Testbed (ADAPT) [42,43], deployed at NASA Ames Research Center, is functionally representative of a spacecraft's electrical power system. Over fifty relays and circuit breakers configure the system into different modes of operation. Therefore, the system behavior is naturally hybrid. Parametric faults, such as changes in resistance and inductance values, can occur in the components. Discrete faults, such as relays becoming stuck, may also occur. Therefore, ADAPT serves as a challenging testbed to verify diagnosis methodologies. ADAPT also allows the injection of certain faults into the hardware, and emulates sensor faults through software spoofing. Further, a simulation testbed of ADAPT has been developed, called Virtual ADAPT, which allows for a richer set of faults to be studied [42, 43].

We demonstrate and experimentally verify our diagnosis approach on ADAPT. We consider a subset of ADAPT that includes a single battery discharging to two DC loads, to experimentally validate our approach. We present hybrid bond graph models of ADAPT components, provide experimental results on ADAPT, and provide detailed simulation experiments investigating the effects of sensor noise and different fault magnitudes on our diagnosis scheme. Some simulation and experimental results have been reported in [36, 37, 86, 133], and preliminary component models have appeared in [116].

First, we present the hybrid bond graph models for the components of the selected configuration, and describe the considered set of faults. Then, we present a formal diagnosability analysis for the system for the selected set of measurements. We next provide simulation results using the Virtual ADAPT simulation testbed. We then describe experimental results obtained from running the diagnoser on the ADAPT hardware.

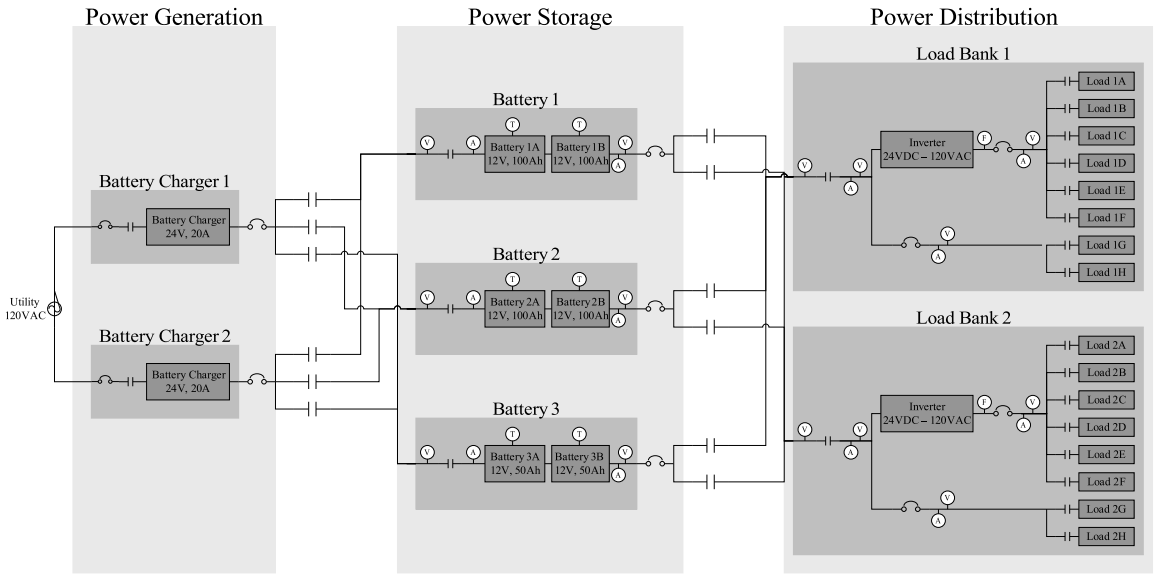


Figure 72: Schematic diagram of ADAPT.

Modeling Electrical Distribution Systems

We demonstrate the proposed diagnosis framework with experiments conducted on the Advanced Diagnostics and Prognostics Testbed (ADAPT) deployed at NASA Ames Research Center [42, 43]. A schematic is shown in Fig. 72. The testbed is functionally representative of a spacecraft's electrical power system, and consists of three subsystems: (*i*) power generation, which includes two battery chargers, (*ii*) power storage, which consists of three sets of lead-acid batteries, and (*iii*) power distribution, which consists of a number of relays and circuit breakers, as well as two inverters and various DC and AC loads.

We consider a subset of ADAPT to demonstrate our approach, which includes a lead-acid battery, two relays, and two DC loads. The electrical circuit equivalent is shown in Fig. 73, and the overall component-based HBG model of the system is shown in Fig. 74. The battery supplies voltage to the relays through a parallel connection, which in turn supply power to the two DC loads. ADAPT contains many sensors, but we choose the minimal set of sensors that our algorithms require. The selected measurements are the battery voltage, $V_B(t)$, and the currents through the relays, $I_{L1}(t)$ and $I_{L2}(t)$, i.e., $M = \{I_{L1}, I_{L2}, V_B\}$. Sensors are also available that observe the state of the switching components, however, to better illustrate the diagnostic power of our approach, we omit them, otherwise diagnosis of discrete faults becomes trivial. In the remainder of the section, we describe the different component models.

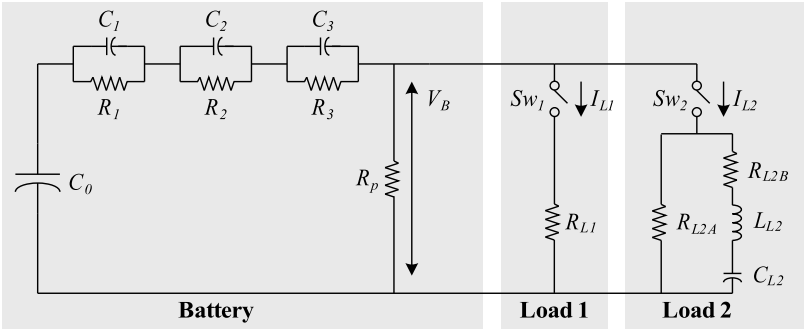


Figure 73: Electrical circuit equivalent for the battery system.

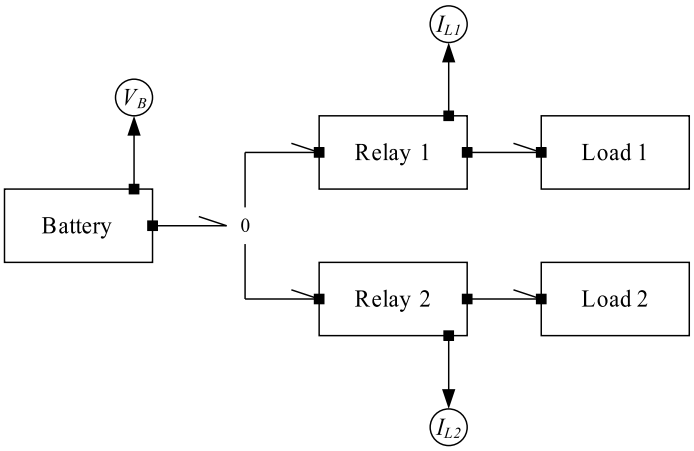


Figure 74: Component-based HBG model of the DC subsystem.

Component Modeling

The battery model describes an electric circuit equivalent based on the model presented in [156,157]. One set of batteries is abstracted to a single battery in the model, so provides a nominal 24 Volts and is rated at 100 Ah. The charge-holding capacity of the battery is modeled by a large capacitance, C_0. The resistance parameters, R_1, R_2, and R_3 are nonlinear functions of battery temperature, θ (which is a function of ambient temperature θ_a and power dissipated through the battery resistances P_B), discharge current, $i(t)$, state of charge, SOC, and depth of charge, DOC (which are functions of $i(t)$ and the charge in C_0, $q(t)$). The equations describing these parameters are given in Table 14 (details can be found in [156]).

The component-oriented bond graph model is given in Fig. 75a. Due to the nonlinearities, the resistances R_1, R_2, and R_2, are modulated elements, denoted by the M in the element type. The parameters vary according to the equations given in Table 14. Essentially, C_0 acts as a large capacitance with a voltage proportional to the amount of charge in the battery. When discharging, the $R-C$ pairs subtract from this voltage and produce the nonlinear curve in the battery output

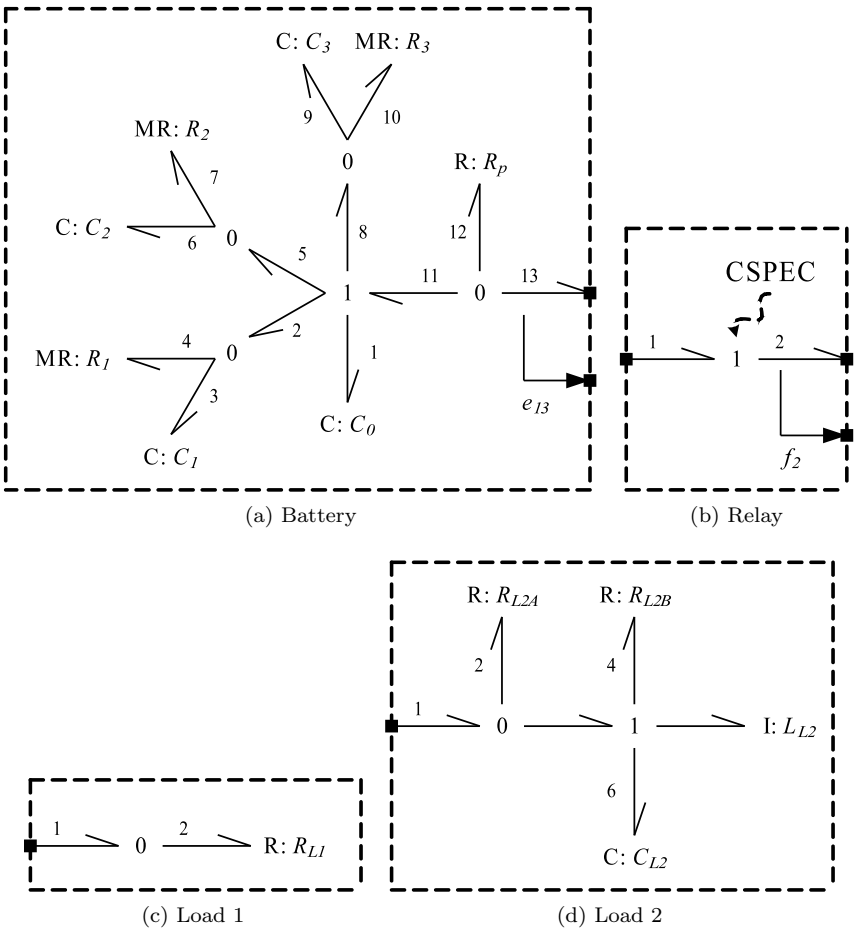


Figure 75: HBG models of the DC loads.

voltage. The R_p resistance models parasitic reactions due to gas emission and other sources of internal energy loss. The variable e_{13} represents the output voltage of the battery.

Relays act as series connections that turn on or off, therefore, they can be modeled by controlled 1-junctions. When off, the junctions inhibit the transfer of energy, and thus disconnect the components attached to the junction. The HBG model is given in Fig. 75b, and the CSPEC given in Fig. 18a can be used for the relays. The flow f_2 can be used to obtain the current through the switching element.

The HBG models for Loads 1 and 2 are given in Figs. 75c and 75d, respectively. Load 1 is simply a resistance, and Load 2 consists of a resistance in parallel with a resistor, inductor, and capacitor in series.

The system parameters were identified using data from different experiments conducted on ADAPT. The identified parameters are given in Table 15.

Table 14: Supplementary Battery Equations

Parameter	Equation
Battery heat power	$P_B = i_{R_p}(t)v_{R_p}(t) + \sum_{j=1}^{3} \lvert i_{R_j}(t)v_{R_j}(t)\rvert$
Battery temperature	$\dot{\theta}(t) = \frac{1}{C_\theta}\left(P_B - \frac{\theta(t)-\theta_a}{R_\theta}\right)$
State of charge	$SOC = 1 - \frac{Q_{max} - q(t)}{C_0^*\left(1-\frac{\theta}{\theta_f}\right)^\epsilon}$
Depth of charge	$DOC = 1 - \frac{Q_{max} - q(t)}{K_c C_0^*\left(1-\frac{\theta}{\theta_f}\right)^\epsilon}\left(1 + (K_c - 1)\left(\frac{i(t)}{I^*}\right)^\delta\right)$
R_1	$R_1 = R_{10} + A_{11}SOC$
R_2	$R_2 = -R_{20}\ln(DOC)$
R_3	$R_3 = \frac{R_{30}\exp A_{31}(1-SOC)}{1+\exp A_{32}i(t)}$

Table 15: Identified System Parameters for the ADAPT Subsystem

Component	Parameters	
Battery	$\theta_a = 22^\circ$C	$C_\theta = 615.3$ Wh/$^\circ$C
	$R_\theta = 0.01^\circ$C/W	$C_0 = 106360$ F
	$C_1 = 51.079$ F	$C_2 = 51.216$ F
	$C_3 = 567.56$ F	$R_{10} = 0.05582\ \Omega$
	$A_{11} = -0.025025$	$R_{20} = 0.001847\ \Omega$
	$R_{30} = 0.3579\ \Omega$	$A_{31} = -2.5315$
	$A_{32} = 0.22208$	$R_p = 500\ \Omega$
	$Q_{max} = 2765360$ C	$K_c = 1.33$
	$C_0^* = 270720$ Ah	$\theta_f = -35^\circ$C
	$\epsilon = 0.642$	$I^* = 5$ A
	$\delta = 0.61$	
Load 1	$R_{L1} = 11.8\ \Omega$	
Load 2	$R_{L2A} = 27.696\ \Omega$	$R_{L2B} = 209.92\ \Omega$
	$C_{L2} = 0.48678$ F	$L_{L2} = 1.9986$ H

Modeling Faults

The complete set of faults is given in Table 16. Common battery faults include loss of charge and resistance increases brought about by battery use and age, which manifest as a side effect of the chemical reactions. Loss of charge is represented by a capacitance decrease, C_0^-, and an increase in internal losses by R_1^+. Faults can occur in the system loads, and these are represented by increases or decreases in their resistance values, R_{L1} and R_{L2A}. For the sensors, we consider bias faults, which produce abrupt changes in the measured values manifesting as constant offsets. Sensor faults are labeled by the measured quantity they represent, e.g., V_B^+ represents a bias fault in the battery voltage sensor. We represent discrete faults in Sw_1 and Sw_2 by fault events α and β, respectively, where a subscript of 0 indicates a stuck-off fault, and a subscript of 1 indicates a stuck-on fault. The complete set of faults is defined as $F = \{C_0^-, R_1^+, R_{L1}^+, R_{L1}^-, R_{L2A}^+, R_{L2A}^-, V_B^+, V_B^-, I_{L1}^+, I_{L1}^-, I_{L2}^+, I_{L2}^-, \alpha_0, \alpha_1, \beta_0, \beta_1\}$.

Table 16: Faults in the ADAPT Model

Fault	Description
C_0^-	Battery capacity decrease
R_1^+	Battery resistance increase
R_{L1}^+	Load 1 resistance increase
R_{L1}^-	Load 1 resistance decrease
R_{L2A}^+	Load 2 resistance increase
R_{L2A}^-	Load 2 resistance decrease
V_B^+	Positive bias in $V_B(t)$ sensor
V_B^-	Negative bias in $V_B(t)$ sensor
I_{L1}^+	Positive bias in $I_{L1}(t)$ sensor
I_{L1}^-	Negative bias in $I_{L1}(t)$ sensor
I_{L2}^+	Positive bias in $I_{L2}(t)$ sensor
I_{L2}^-	Negative bias in $I_{L2}(t)$ sensor
α_0	Sw_1 stuck-off
α_1	Sw_1 stuck-on
β_0	Sw_2 stuck-off
β_1	Sw_2 stuck-on

The selected set of faults comprises a varied set of faults in both sensors, actuators, and the physical process, as well as both parametric and discrete faults. Therefore, the selected faults are representative enough to demonstrate our diagnosis approach.

ADAPT consists of three operator roles, a User, which simulates a crew member, an Observer, which logs all system data, and an Antagonist, which injects faults into the system. Through the Antagonist functionality, various faults can be injected through software. Sensor faults can be injected through online spoofing capabilities. Discrete faults can be injected by blocking control commands or injecting spurious commands. Additionally, faults can be injected manually by changing settings of the loads or switching components on or off. With this combined functionality, we can inject $\{R_{L1}^+, R_{L1}^-, V_B^+, V_B^-, I_{L1}^+, I_{L1}^-, I_{L2}^+, I_{L2}^-, \alpha_0, \alpha_1, \beta_0, \beta_1\}$ on the testbed. The remaining faults, $\{C_0^-, R_1^+, R_{L2A}^+, R_{L2A}^-, \}$, must be injected in the simulation testbed and therefore can only be studied in simulation.

Diagnosability Analysis

Each relay has four CSPEC states, with two nominal states, on and off, and two fault states, stuck-on and stuck-off. Therefore, there are 16 possible system modes, which consists of 4 nominal modes and 12 fault modes. We denote the system mode as q_{ij}, where i is the mode of Sw_1, and j is the mode of Sw_2. For example, $q_{1\beta_0}$ is the mode where Sw_1 is on and Sw_2 is stuck off. We define the complete set of modes as $Q = Q_N \cup Q_F$, with the set of nominal modes given as $Q_N = \{q_{00}, q_{01}, \ldots, q_{11}\}$, and the set of fault modes given as $Q_F = \{q_{\alpha_0 0}, q_{\alpha_0 1}, q_{\alpha_1 0}, q_{\alpha_1 1}, q_{0\beta_0}, q_{0\beta_1}, q_{1\beta_0}, q_{1\beta_1}, q_{\alpha_0\beta_0}, q_{\alpha_0\beta_1}, q_{\alpha_1\beta_0}, q_{\alpha_1\beta_1}\}$. We allow controlled mode events that switch the system from any one

Table 17: Fault Signatures and Relative Measurement Orderings for the ADAPT Subsystem

Fault	V_B	I_{L1}	I_{L2}	Measurement Orderings
(V_B^+, q_{**})	+0,X	00,X	00,X	$V_B \prec I_{L1}, V_B \prec I_{L2}$
(V_B^-, q_{**})	-0,X	00,X	00,X	$V_B \prec I_{L1}, V_B \prec I_{L2}$
(I_{L1}^+, q_{**})	00,X	+0,X	00,X	$I_{L1} \prec V_B, I_{L1} \prec I_{L2}$
(I_{L1}^-, q_{**})	00,X	-0,X	00,X	$I_{L1} \prec V_B, I_{L1} \prec I_{L2}$
(I_{L2}^+, q_{**})	00,X	00,X	+0,X	$I_{L1} \prec V_B, I_{L2} \prec I_{L1}$
(I_{L2}^-, q_{**})	00,X	00,X	-0,X	$I_{L1} \prec V_B, I_{L2} \prec I_{L1}$
(C_0^-, q_{11})	+-,X	+-,X	+-,X	$\varnothing$
(R_1^+, q_{11})	0-,X	0-,X	0-,X	$\varnothing$
(R_{L1}^+, q_{11})	0*,X	-+,X	0*,X	$I_{L1} \prec V_B, I_{L1} \prec I_{L2}$
(R_{L1}^-, q_{11})	0*,X	+-,X	0*,X	$I_{L1} \prec V_B, I_{L1} \prec I_{L2}$
(R_{L2A}^+, q_{11})	0*,X	0*,X	-+,X	$I_{L2} \prec V_B, I_{L2} \prec I_{L1}$
(R_{L2A}^-, q_{11})	0*,X	0*,X	+-,X	$I_{L2} \prec V_B, I_{L2} \prec I_{L1}$
$(\alpha_0, q_{\alpha_0 1})$	0*,X	-*,Z	0*,X	$I_{L1} \prec V_B, I_{L1} \prec I_{L2}$
$(\alpha_1, q_{\alpha_1 1})$	0*,X	+*,N	0*,X	$I_{L1} \prec V_B, I_{L1} \prec I_{L2}$
$(\beta_0, q_{1\beta_0})$	0*,X	0*,X	-*,Z	$I_{L2} \prec V_B, I_{L2} \prec I_{L1}$
$(\beta_1, q_{1\beta_1})$	0*,X	0*,X	+*,N	$I_{L2} \prec V_B, I_{L2} \prec I_{L1}$

controlled mode to another, i.e., $\Sigma_Q = \{\sigma_{q_{00}}, \sigma_{q_{01}}, \ldots, \sigma_{q_{11}}\}$. We only restrict the occurrence of controlled mode changes by not allowing controlled mode changes to the current expected controlled mode, e.g., $\sigma_{q_{01}}$ is not allowed if the expected mode is q_{01}. We also restrict discrete faults to only occurring from expected modes where a deviation would be produced, e.g., α_1 would not produce any deviations if it occurred in a mode where Sw_1 was already on. With F, M, and Q defined, we can now perform diagnosability analysis of the system.

The fault signatures and relative measurement orderings for the above faults are given in Table 17 for selected modes (q_{**} indicates the signatures and orderings are valid for any mode). The nonlinearities in the battery introduce ambiguity in the qualitative signatures, and this is denoted by the * symbol. For parametric faults for which prediction yields +*,X or -*,X, only +-,X and -+,X are possible, because ++,X and --,X imply an unstable system. For example, for R_{L1}^+, the prediction for I_{L1} is -*,X, but we can restrict the signature to -+,X, as shown in Table 17. A signature of 0*, however, may manifest as 0+ or 0-. All possible effects must be included in the fault models. Also note that since the sensors are not part of feedback loops in the system, sensor faults affect only the measurement provided by the sensor. The other measurements are not affected, and so the corresponding fault signatures are denoted by 00, indicating no change in the measurement from expected behavior.

Example. Selected fault models for ADAPT are shown in Fig. 76. Consider the fault model $\mathcal{L}_{C_0^-, q_{11}}$, shown in Fig. 76a. No orderings can be derived, since discontinuities are produced on all measurements, and so if all measurements deviate at the same time they can be taken in any order.

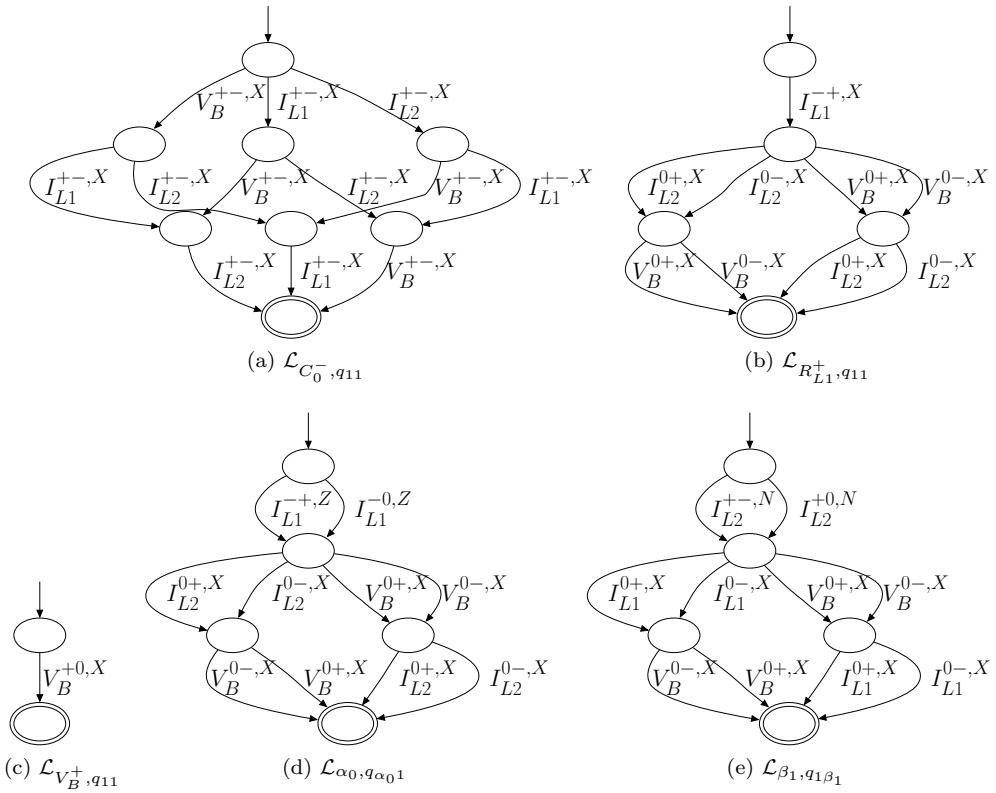


Figure 76: Selected fault models for ADAPT.

Therefore the fault model must account for all possible orders of measurement deviations. Any sequence variation of the fault trace $V_B^{+-} I_{L1}^{+-} I_{L2}^{+-}$ is accepted.

Since controlled mode changes occur infrequently and almost all faults produce discontinuities, we take $n_{back} = 0$, i.e., the true mode of fault occurrence, i.e., the mode immediately before the fault occurs, is the expected mode of the system at the time of fault detection. Given any one mode, the system is 1-fault diagnosable for that mode. After at most two measurement deviations, a unique candidate can be isolated. However, over all modes, the system is not diagnosable. Fig. 77 gives a partial diagnoser for the system that illustrates this property, with $F = \{C_0^-, R_{L1}^-\}$ and initial mode q_{11} with $\sigma_{q_{01}}$ being the only controlled mode change event. We can see that if $I_{L1}^{+-,X}\sigma_{q_{01}}$ occurs, we reach an accepting state that corresponds to a diagnosis with multiple candidates. After that event, both C_0^- and R_{L1}^+ are consistent. Since the state is accepting, it is possible that no new measurement deviations will occur to distinguish the faults. In mode q_{01}, R_{L1}^- cannot produce changes on the measurements, i.e., the fault model for (R_{L1}, q_{01}) consists of a single accepting state, with no events. So, the trace is a maximal candidate trace for (R_{L1}^-, q_{01}). If R_{L1}^+ is the true fault in this scenario, we would have to wait infinitely long to verify it. Therefore, the system is not diagnosable.

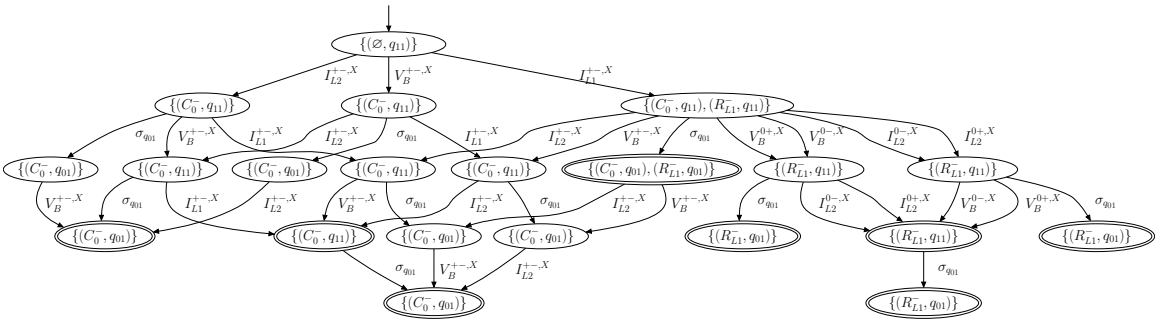


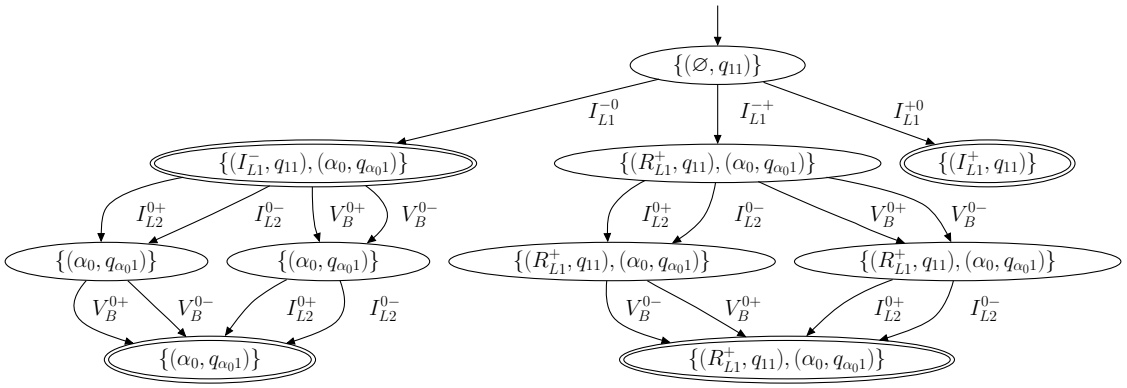
Figure 77: Hybrid diagnoser for $F = \{C_0^-, R_{L1}^-\}$ and initial mode q_{11} with $l = 1$.

We can see, however, that the system is, in fact, Q-diagnosable. If we prevent $\sigma_{q_{01}}$ from occurring, or change back to q_{11} if it does occur, more measurements will deviate and we can distinguish the candidate uniquely. The diagnoser shows for which observed traces controlled mode changes should be blocked or executed to improve distinguishability.

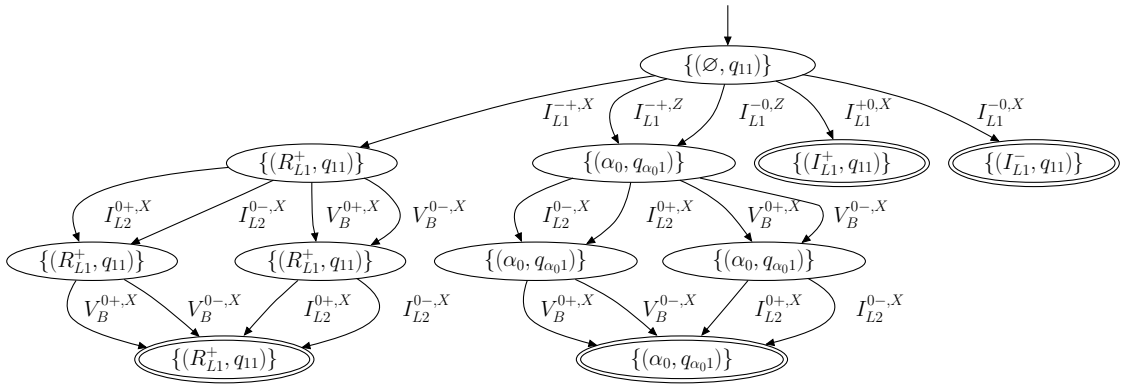
The diagnosability analysis also reveals that the discrete change feature is necessary to distinguish certain pairs of faults. This is illustrated in Fig. 78, with mode change events removed for clarity. Without the discrete change feature (Fig. 78a), we may reach an accepting state when, for example, I_{L1}^{-0} is observed, signifying that if we do not observe any future measurement deviations, we cannot obtain a unique diagnosis. Or, if I_{L1}^{-+} is observed, we cannot distinguish between the load fault and the discrete fault. However, with the discrete change feature (Fig. 78b), we can differentiate between the faults with a single measurement deviation. The discrete change feature significantly increases the diagnosability of the system, and enables quick diagnosis of discrete faults.

Since the system is not 1-fault diagnosable, it is not 2-fault diagnosable. Even if we disallow controlled mode changes during isolation, however, it is still not 2-fault diagnosable. For example, if the trace $I_{L1}^{+-,X} I_{L2}^{+-,X} V_B^{0-,X}$ is observed in mode q_{11}, it could be attributed to many candidates. The path of the diagnoser corresponding to this trace is shown in Fig. 79. After $I_{L1}^{+-,X}$, the (truncated) hypothesis set and diagnosis are both $\{(C_0^-, q_{11}), (R_{L1}^-, q_{11})\}$. After $I_{L2}^{+-,X}$, the hypothesis set is $\{(C_0^-, q_{11}), (R_{L2A}^-, q_{11})\}$, so the minimal diagnosis is $\{(C_0^-, q_{11}), (R_{L1}^- R_{L2A}^-, q_{11})\}$. After $V_B^{0-,X}$, the hypothesis set $\{(R_1^-, q_{11}), (R_{L1}^+, q_{11}), (R_{L1}^-, q_{11}), (R_{L2A}^+, q_{11}), (R_{L2A}^-, q_{11})), (\alpha_0, q_{\alpha_0 1}), (\beta_0, q_{1\beta_0})\}$, due to the ambiguity for the V_B measurement, which results in the minimal diagnosis shown in the figure.

Example. To illustrate what happens for multiple discrete faults, we provide a partial diagnoser with $F = \{\alpha_1, \beta_1\}$ for initial mode q_{00} and $l = 2$, shown in Fig. 80. Controlled mode changes are omitted for clarity and space. Consider the fault trace $I_{L1}^{+-,N} I_{L2}^{+-,N} V_B^{0+,X}$. After $I_{L1}^{+-,N}$, the hypothesis set is $\{(\alpha_1, q_{\alpha_1 0}), (\alpha_1, q_{\alpha_1 1}))\}$ and the initial diagnosis is only $\{(\alpha_1, q_{\alpha_1 0})\}$ since the initial mode



(a) Without the discrete change feature.



(b) With the discrete change feature.

Figure 78: Hybrid diagnoser for $F = \{I_{L1}^{+}, I_{L1}^{-}, R_{L1}^{+}, \alpha_0, \}$ with initial mode q_{11} and $l = 1$.

was q_{00}. After $I_{L2}^{+-,N}$, the hypothesis set is $\{(\beta_1, q_{0\beta_1}), (\beta_1, q_{1\beta_1}), (\beta_1, q_{\alpha_1\beta_1})\}$, so the new diagnosis is $\{(\alpha_1\beta_1, q_{\alpha_1\beta_1})\}$. After $V_B^{0+,X}$, the hypothesis set is $\{(\alpha_1, q_{\alpha_1 0}), (\alpha_1, q_{\alpha_1 1}), (\beta_1, q_{0\beta_1}), (\beta_1, q_{1\beta_1})\}$, so the diagnosis remains the same. Note that in online diagnosis, the hypothesis sets would contain only candidates whose modes can be reached with no event or an unobservable fault event from the current candidates.

Table 18 shows the size of the diagnosers for $l = 1$, $l = 2$, and $l = 3$ for the system, for all faults and all modes. As discussed in Chapter VII, the complexity is bounded by the number of measurements and the number of modes. With 3 measurements and 8 nominal modes, there are 29 possible events. The upper bound is $(8 + 3)!$, which is approximately 4×10^7. Due to the constraints imposed by relative measurement orderings and the mode transition structure, the diagnosers, although large, are still well below this upper bound. The increase in complexity from $l = 2$ to $l = 3$ is not as large as that for $l = 1$ to $l = 2$, because distinguishability decreases when candidate cardinality increases, and therefore there are more event sequences that can be explained by candidates of smaller cardinality. In other words, there are proportionately less new traces added

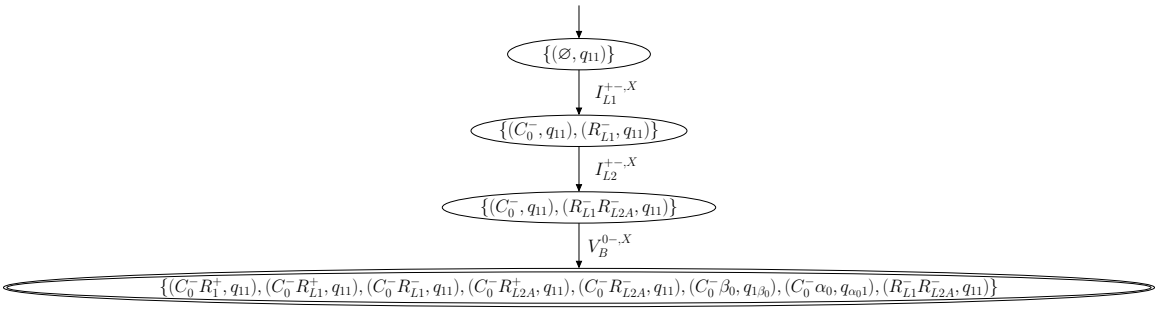


Figure 79: Hybrid diagnoser path for initial mode q_{11} with $l = 2$.

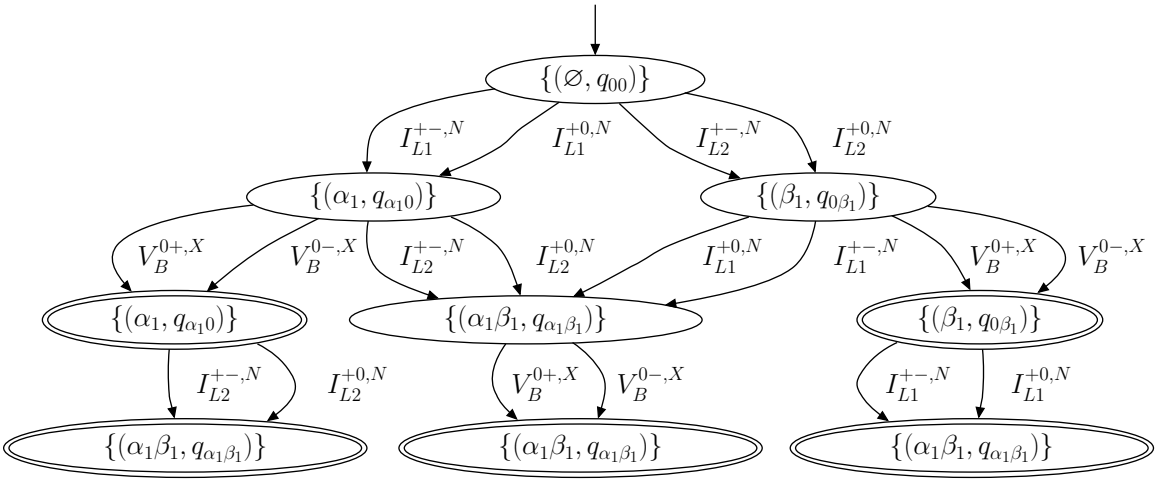


Figure 80: Hybrid diagnoser for $F = \{\alpha_1, \beta_1\}$ and initial mode q_{00} with $l = 2$.

when going from $l = 2$ to $l = 3$ then from $l = 1$ to $l = 2$. It is also clear from these results that the global diagnoser implementation is not as feasible for hybrid systems due to the large space requirements, however, they are needed to determine diagnosability of the system.

The pruned versions of the diagnosers also show considerable improvements over the nonpruned versions. The reduction is greater for the $l = 1$ diagnoser, because unique diagnoses are obtained earlier. With multiple faults, it is very likely that a new measurement deviation will bring about a new minimal diagnosis, whereas with single faults, diagnoses only become smaller, so once a unique diagnosis is known, future events from that state are unnecessary. The proportion of the states that are accepting also increases with pruning, because pruning increases the proportion of leaf states, and leaf states in the pruned diagnoser all become accepting states.

Simulation Results

The accuracy of our diagnosis approach is critically dependent on the fault detection and symbol generation processes. Due to model imperfections and sensor noise, the fault detectors have to be

Table 18: Diagnoser Design Results for the ADAPT Subsystem

Parameter	$\mathcal{D}^{1*}_{F,M,Q}$		$\mathcal{D}^{2*}_{F,M,Q}$		$\mathcal{D}^{3*}_{F,M,Q}$	
	Not Pruned	Pruned	Not Pruned	Pruned	Not Pruned	Pruned
States	488	83	4,748	2,658	12,119	7,707
Accepting States	350	75	4,308	2,587	11,679	7,632
Events	29	29	29	29	29	29
Transitions	1,784	262	17,518	8,892	43,667	24,651

tuned to minimize missed detections (false negatives), and false alarms (false positives). A trade-off exists between these two, because a more sensitive fault detector will get more false positives, but fewer false negatives. Similarly, a less sensitive fault detector will get more false negatives, but less false positives. In our experiments, the fault detectors were empirically tuned to the highest possible sensitivity that avoided false alarms for the observed levels of noise under nominal conditions. To study the performance of the diagnosis algorithms under different fault and noise conditions, we need to perform a large number of experiments. In addition to experiments from the actual testbed, we ran simulation experiments on the VIRTUAL ADAPT testbed [42,43,109]. Simulation also allows us to introduce faults that cannot be injected into the actual system in a safe manner.

In the following simulation experiments, we considered different fault magnitudes and different levels of sensor noise to investigate the robustness and sensitivity of our fault detection and isolation scheme. We used a zero-mean Gaussian noise model, and the noise level was reflected in the variance. The three noise levels (N_0, N_1, N_2) reflect no noise, the observed noise magnitudes of the testbed, and double the observed noise. These values were selected as 0, 4×10^{-4}, and 8×10^{-4} for the voltage sensor, and 0, 4×10^{-5}, and 8×10^{-5} for the current sensors.

The diagnosis results are summarized in Table 19. For the sensor faults, the magnitude is given as an additive bias in V or A. For the parametric faults, the faulty parameter value is given by its nominal value multiplied by the given factor, e.g., a factor of 1.10 increases R_{L1} by 10% of its nominal value of 11.8 Ω. Since discrete faults do not have magnitude, we investigate two possibilities for each, whether the switch became stuck (i.e., ignored a switching command), or switched unexpectedly (i.e., it switched without a switching command). Ten experiments were performed for each fault, magnitude, and noise level. The table presents the average results over these runs. In each of the scenarios, the time of fault injection, t_f, was set at 500.0 s. The times for detection and isolation are denoted by t_d and t_i, respectively. In some cases the true fault cannot be uniquely isolated, so t_i represents the time at which the fault candidate list stopped reducing. We report on the average times to detect and isolate, the average size of the final fault candidate list F_n, and the percentage of times the true fault was in F_n.

Table 19: ADAPT Experiments with Different Fault Magnitudes and Noise Levels

Fault	Magnitude	$t_d - t_f$ (s)			$t_i - t_f$ (s)			$\lvert F_n \rvert$			$f \in F_n$ (%)		
		N_0	N_1	N_2	N_0	N_1	N_2	N_0	N_1	N_2	N_0	N_1	N_2
V_B^+	+ 0.10	0.50	1.20	1.95	0.50	1.20	3.65	2.00	1.40	1.80	100	70	80
	+ 0.20	0.00	0.40	0.80	0.00	0.40	1.45	2.00	2.00	1.90	100	100	90
	+ 0.40	0.00	0.00	0.35	0.00	0.00	1.65	2.00	2.00	1.80	100	100	80
V_B^-	− 0.10	0.50	1.25	1.80	17.00	17.75	18.30	1.00	1.00	1.00	100	100	100
	− 0.20	0.00	0.50	0.95	16.50	17.00	21.00	1.00	1.00	0.90	100	100	80
	− 0.40	0.00	0.05	0.30	16.50	16.55	14.8	1.00	1.00	1.00	100	100	80
I_{L1}^+	+ 0.10	0.50	0.20	0.45	0.50	0.20	0.45	3.00	3.00	3.00	100	100	100
	+ 0.20	0.00	0.00	0.00	0.00	0.00	0.00	3.00	3.00	3.00	100	100	100
	+ 0.40	0.00	0.00	0.00	0.00	0.00	0.00	3.00	3.00	3.00	100	100	100
I_{L1}^-	− 0.10	0.00	0.35	0.50	17.50	17.85	18.00	2.00	2.00	2.00	100	100	100
	− 0.20	0.00	0.00	0.00	17.50	17.50	17.50	2.00	2.00	2.00	100	100	100
	− 0.40	0.00	0.00	0.00	17.50	17.50	17.50	2.00	2.00	2.00	100	100	100
I_{L2}^+	+ 0.10	0.50	0.20	0.45	18.00	17.70	17.95	3.00	3.00	3.00	100	100	100
	+ 0.20	0.00	0.00	0.00	17.50	17.50	17.50	3.00	3.00	3.00	100	100	100
	+ 0.40	0.00	0.00	0.00	17.50	17.50	17.50	3.00	3.00	3.00	100	100	100
I_{L2}^-	− 0.10	0.00	0.35	0.50	17.50	17.85	18.00	2.00	2.00	2.00	100	100	100
	− 0.20	0.00	0.00	0.00	17.50	17.50	17.50	2.00	2.00	2.00	100	100	100
	− 0.40	0.00	0.00	0.00	17.50	17.50	17.50	2.00	2.00	2.00	100	100	100
C_0^-	× 0.99	0.00	0.45	0.60	2.00	2.80	13.00	1.00	1.00	1.00	100	100	100
	× 0.98	0.00	0.00	0.10	1.00	1.00	1.05	1.00	1.00	1.00	100	100	100
	× 0.96	0.00	0.00	0.00	0.50	0.50	0.55	1.00	1.00	1.00	100	100	100
R_1^+	× 2.00	2.00	4.05	7.20	4.50	15.25	22.05	1.00	1.00	1.00	100	60	30
	× 2.20	2.00	3.55	5.00	4.50	10.65	17.50	1.00	1.00	1.00	100	100	90
	× 2.40	2.00	3.05	4.75	4.00	8.25	11.75	1.00	1.00	1.00	100	100	100
R_{L1}^+	× 1.05	0.50	0.30	0.50	18.00	17.80	18.00	2.00	2.00	2.00	100	100	100
	× 1.10	0.00	0.00	0.00	17.50	17.50	17.50	2.00	2.00	2.00	100	100	100
	× 1.20	0.00	0.00	0.00	8.50	17.50	17.50	1.00	2.00	2.00	100	100	100
R_{L1}^-	× 0.95	0.00	0.05	0.40	0.00	0.05	0.4	3.00	3.00	3.00	100	100	100
	× 0.90	0.00	0.00	0.00	0.00	0.00	0.00	3.00	3.00	3.00	100	100	100
	× 0.80	0.00	0.00	0.00	3.50	0.00	0.00	1.00	3.00	3.00	100	100	100
R_{L2A}^+	× 1.05	1.00	0.95	1.40	18.50	18.45	18.90	2.00	2.00	2.00	100	100	100
	× 1.10	0.50	0.50	0.50	18.00	18.00	18.00	2.00	2.00	2.00	100	100	100
	× 1.20	0.00	0.00	0.10	17.50	17.50	17.60	2.00	2.00	2.00	100	100	100
R_{L2A}^-	× 0.95	1.00	0.90	0.95	18.50	18.40	18.45	3.00	3.00	3.00	100	100	100
	× 0.90	0.50	0.15	0.45	18.00	17.65	17.95	3.00	3.00	3.00	100	100	100
	× 0.80	0.00	0.00	0.00	17.50	17.50	17.50	3.00	3.00	3.00	100	100	100
α_0	stuck	2.00	2.00	2.00	5.00	5.00	5.00	1.00	1.00	1.00	100	100	100
	unexpected	0.00	0.00	0.00	3.00	3.00	3.00	1.00	1.00	1.00	100	100	100
α_1	stuck	2.00	2.00	2.00	2.00	2.70	4.60	1.00	1.00	1.00	100	100	100
	unexpected	0.00	0.00	0.00	1.50	2.75	3.00	1.00	1.00	1.00	100	100	100
β_0	stuck	2.00	2.00	2.00	5.00	5.00	5.00	1.00	1.00	1.00	100	100	100
	unexpected	0.00	0.00	0.00	3.00	3.00	3.00	1.00	1.00	1.00	100	100	100
β_1	stuck	2.00	2.00	2.00	2.50	5.00	5.00	1.00	1.00	1.00	100	100	100
	unexpected	0.00	0.00	0.00	3.00	3.00	3.00	1.00	1.00	1.00	100	100	100

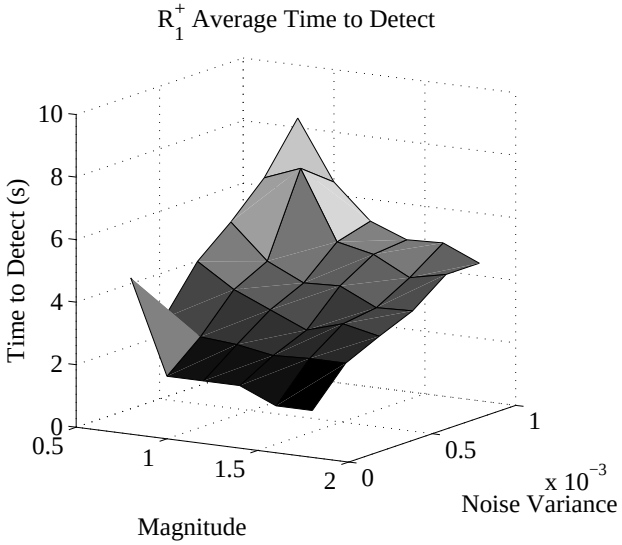


Figure 81: Average time to detect for R_1^+ with varying noise variance and fault magnitude.

The results show that the sensor noise and fault magnitude can have a significant effect on time to fault detection. Fig. 81 shows the average time to detect as a function of the variance in the sensor noise and the fault magnitude for R_1^+. For smaller fault magnitudes and a lower signal to noise ratio, it takes longer for the effects of the fault to be identified in relation to the noise band. Therefore, reliable detection takes longer. As shown in Table 19, faults are detected faster when magnitudes are larger, because a shorter interval is needed to determine that the mean of the residual is statistically outside of the computed signal variance. Fault detection times also improve with lower noise, because the deviations caused by a fault are more clearly differentiated from the noise. Similar results were obtained for the other faults.

Sensor noise and fault magnitude can also affect the isolation results. If an incorrect symbol is generated, then the true fault may be eliminated as a candidate. This situation is shown well by the experiments with R_1^+. Fig. 82 shows the isolation rates for this fault as a function of fault magnitude and sensor noise. When the fault magnitude was large enough, the symbols were correctly generated and the fault correctly isolated, even for the highest levels of noise. However, as fault magnitude decreased and noise increased, the wrong fault was sometimes isolated. R_1^+ produces a first-order change on the voltage. If this change is detected early, then the transient can be observed and the signature correctly computed as `0-`. If the change is detected after the initial transient, when the voltage has already reached a steady state, then the slope is observed to be zero, so `-0` is computed, thus isolating V_B^- as the fault. The incorrect signature was more likely to be computed when the fault magnitude was low and sensor noise was high, because detection of the change in $V_B(t)$ is more

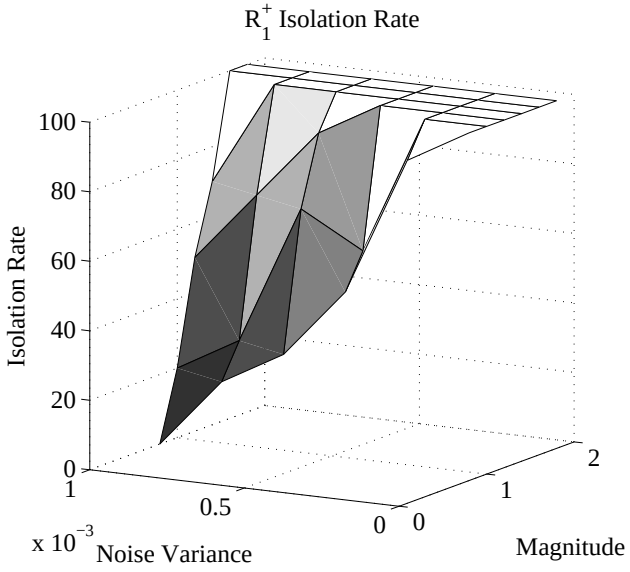


Figure 82: Isolation rate for R_1^+ with varying noise variance and fault magnitude. The fault is considered successfully isolated if it is in the final list of faults returned by the diagnoser.

likely to occur after the transient. Knowledge of expected fault magnitudes and noise levels can help tune the fault detector parameters to correctly compute these features. For the other faults, this problem did not occur except for a few cases with V_B^+ and V_B^- when high noise caused the slope computation to be `+` or `-` instead of `0`, which can be viewed as a false alarm in the slope computation.

We have also performed simulation experiments for multiple faults. As an example, consider α_0 occurring at 500.0 s followed by β_0 at 501.0 s. The measured and estimated outputs are shown in Fig. 83. In this case, it is easy to determine that both faults have occurred. At 500.0 s, a fault is detected by an decrease in $I_{L1}(t)$, so the initial diagnosis is $\{(R_1^+, q_{11}), (R_{L1}^+, q_{11}), (I_{L1}^-, q_{11}), (\alpha_0, q_{\alpha_0 1})\}$. At 501.0 s, a decrease is detected in $I_{L2}(t)$, so the new minimal diagnosis is $\{(R_1^+, q_{11})$, (R_{L1}^+, q_{11}), $(\alpha_0, q_{\alpha_0 1})\}$. At 501.5 s, an increase in $V_B(t)$ is detected, so the new minimal diagnosis is $\{(R_{L1}^+, q_{11})$, $(\alpha_0, q_{\alpha_0 1})\}$. At 503.0 s, the symbol generator determines that $I_{L1}(t)$ went to zero, so the minimal diagnosis is now $\{(\alpha_0, q_{\alpha_0 1})\}$. At 503.5 s, the slope of $V_B(t)$ is computed as `+`, therefore the full signature of $V_B(t)$ is known to be `0+`. At 504.0 s, the symbol generator determines that $I_{L2}(t)$ also went to zero, so now α_0 by itself is no longer consistent, and this results in a new minimal diagnosis of $\{(\alpha_0\beta_0, q_{\alpha_0\beta_0})\}$. At this point, a unique double fault is known, so for $l = 2$, fault isolation can terminate. The symbol generator later determines that $V_B(t)$ exhibited no discrete change, and computes a `0` slope for the currents, classifying the changes as discontinuities, for which the diagnosis is still consistent.

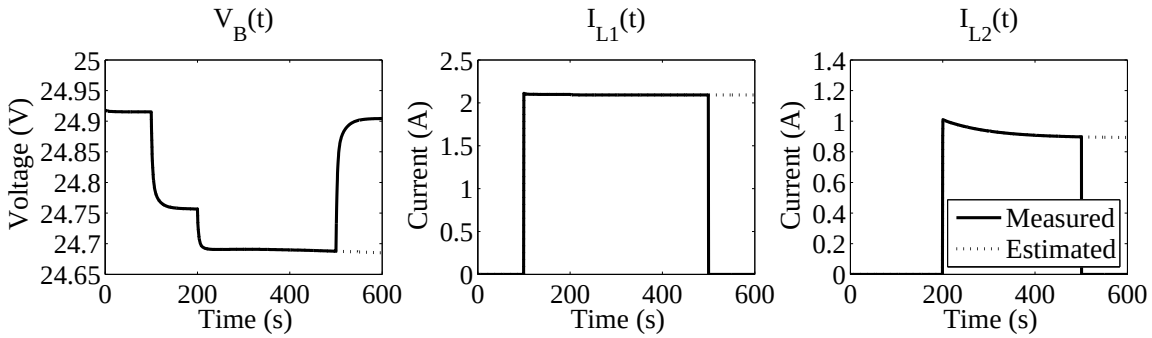


Figure 83: α_0 fault at 500.0 s and β_0 fault at 501.0 s.

To illustrate the issues in multiple fault diagnosis, we consider R_{L1}^- and $R_{L2A}-$ occurring together at 500.0 s, and omit the sensor noise. We varied the fault magnitude of each, and observed how fault masking affected the diagnosis results. Fig. 84 illustrates the role of fault magnitude. Unless one of the fault magnitudes was very small, the diagnoser correctly determined that both faults had occurred. In this case, faults in the loads only noticeably affect the current in the other load if their magnitude is large. For example, Fig. 85 shows the plots for when R_{L1} decreased by 80%, and R_{L2A} decreased by only 2.5%. In this case, the increase in $I_{L2}(t)$ due to R_{L2A}^- is visible, but too small to be detected. The decrease in $I_{L2}(t)$ due to R_{L1}^- masks the change, and therefore only R_{L1}^- is isolated. If the compensation is not great enough, then the increase on $I_{L2}(t)$ due to R_{L2A}^- is detected along with the increase on $I_{L1}(t)$ due to R_{L1}^-, therefore, both faults are isolated. For the case where R_{L1} decreased by only 1%, and R_{L2A} decreased by 80%, the change in $I_{L1}(t)$ due to R_{L1}^- is compensated for by R_{L2A}^-, and therefore only R_{L2A}^- is isolated.

Experimental Results

We have performed online experiments on the ADAPT testbed for single fault diagnosis, i.e., $l = 1$. In online experiments, we have to cope with model uncertainty in addition to sensor noise. To demonstrate the diagnosis approach, we show the results obtained for load faults, switch faults, and a sensor fault. The measurements were sampled at 2 Hz for all the experiments. The mean window, W_1, was selected as 5 samples, the slope window, W_2, as 15 samples, and the discrete change window W_3, as 5 samples. Because not all symbols are available immediately (e.g., the discrete change symbol is available only after 5 samples), we trace multiple paths in the diagnoser until all symbols for a measurement are known, and then stop tracing paths that become inconsistent.

The nominal behavior of the system in shown in Fig. 86. As shown in the figure, the system parameters are fairly accurate, and the observer tracks well. For the parametric faults, Load 1 is

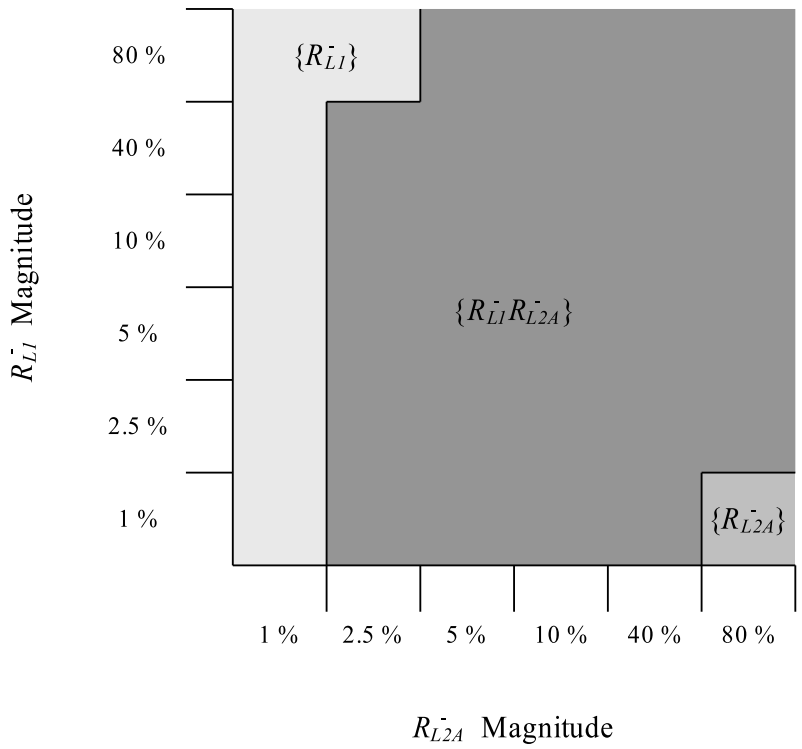


Figure 84: Role of fault magnitude in multiple fault isolation.

first brought online, followed by Load 2, and the fault is injected in mode q_{11}. For discrete faults, we investigate faults in Sw_1, so the fault is injected in mode q_{11} or q_{01}.

For the first experiment, a 33% decrease in the Load 1 resistance, R_{L1}^-, is manually injected at 653.0 s in mode q_{11}, by abruptly changing the resistance setting on the load. The measured and estimated outputs are shown in Fig. 87. A partial diagnoser is given in Fig. 88. The decrease in resistance increases the current drawn by the load abruptly, and this change is detected at 653.5 s. Since the slope of the change is not yet known, the possible fault hypotheses are

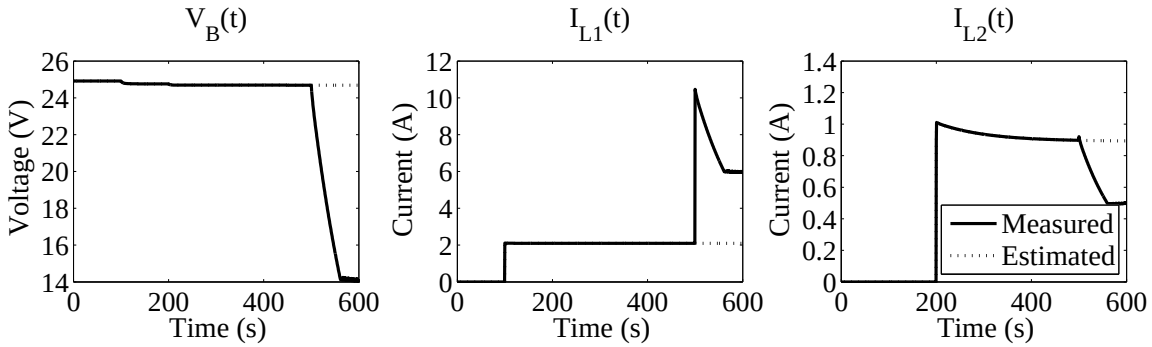


Figure 85: R_{L1}^- at 80% and R_{L2A}^- at 2.5% occurring simultaneously at 500.0 s.

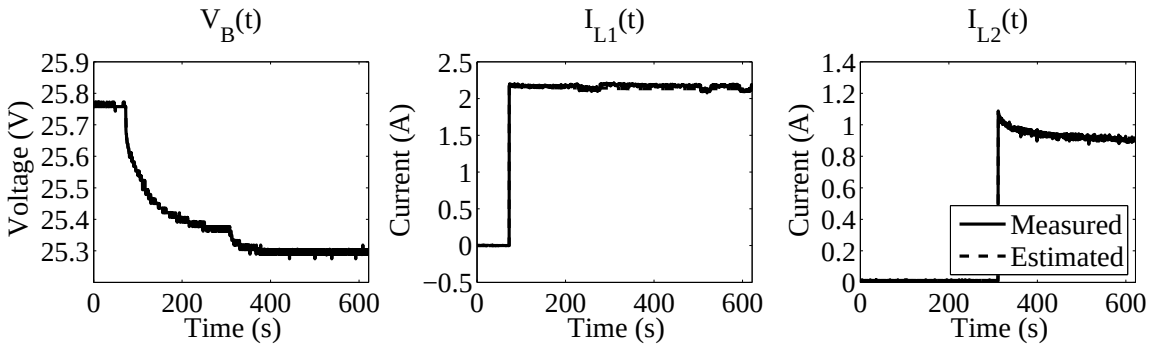


Figure 86: Nominal system operation.

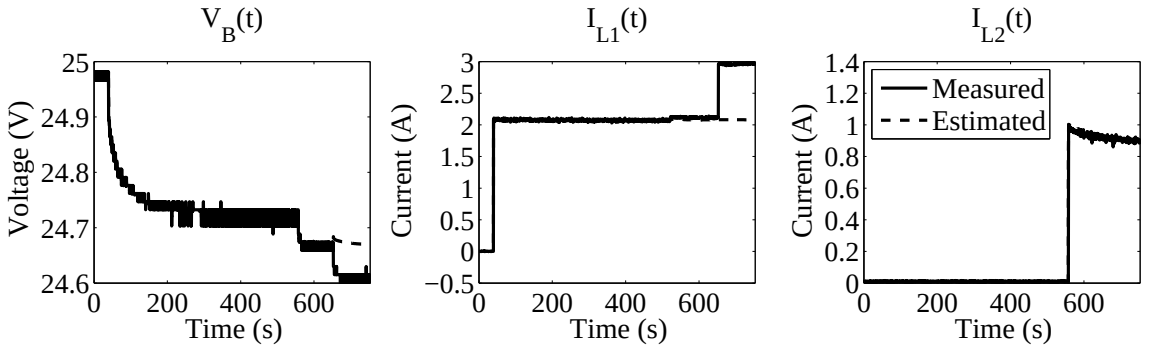


Figure 87: R_{L1}^- fault, where R_{L1} decreases by 33%.

$\{(C_0^-, q_{11}), (R_{L1}^-, q_{11}), (I_{L1}^+, q_{11})\}$. Faults R_{L2A}^+ and R_{L2A}^- are not included, because even though they may cause the current to increase, measurement orderings predict that I_{L2} would have deviated first instead. Also, α_1 is not included, because if it were to occur in q_{11}, we would not have observed a deviation. At 655.0 s, a decrease is detected in $V_B(t)$. Since I_{L1}^+ cannot affect $V_B(t)$, it is dropped. C_0^- is also dropped because it would have increased, and not decreased, the battery voltage. Therefore the fault is isolated as R_{L1}^-.

For a second scenario, a 100% increase in the Load 1 resistance, R_{L1}^+, is manually injected at 439.5 s in mode q_{11}. The measured and estimated outputs are shown in Fig. 89. A partial diagnoser is given in Fig. 90. The increase in resistance causes a discontinuous drop in the current, detected at 440.0 s. Since the slope has not yet been computed, the possible fault candidates are $\{(R_1^+, q_{11}), (R_{L1}^+, q_{11}), (I_{L1}^-, q_{11}), (\alpha_0, q_{\alpha_0 1})\}$. At 441.0 s, an increase is detected in $V_B(t)$. Since I_{L1}^- cannot affect $V_B(t)$, it is dropped. R_1^+ is also dropped because it would have decreased, and not increased, the battery voltage. At 442.5 s, the it is determined that no discrete change in $I_{L1}(t)$ occurred, so R_{L1}^+ is isolated as the true fault.

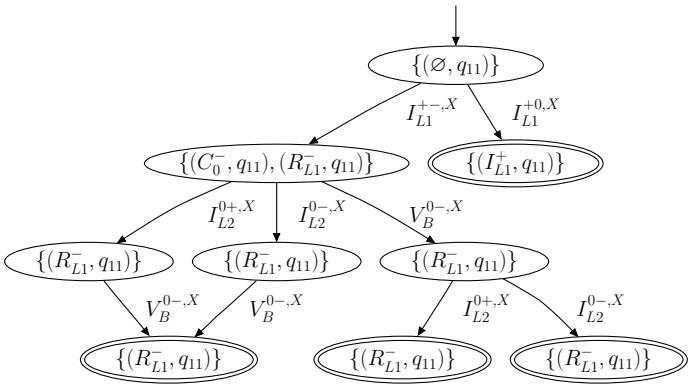


Figure 88: Partial diagnoser for isolating R_{L1}^-.

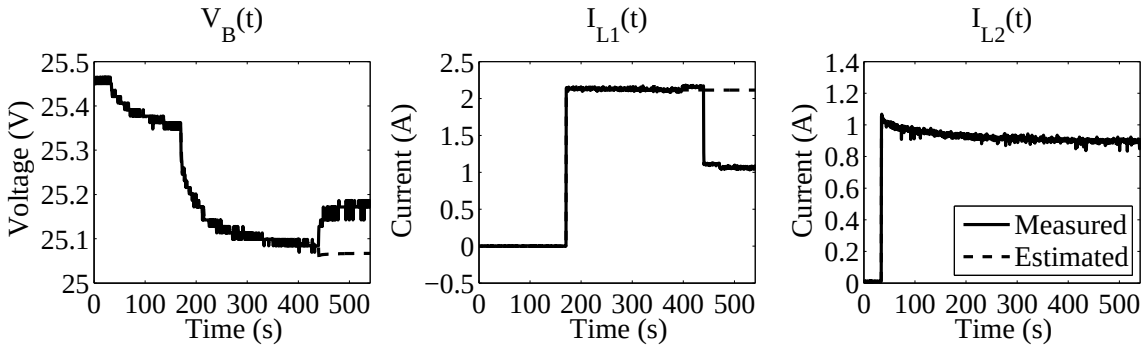


Figure 89: R_{L1}^+ fault, where R_{L1} increases by 100%.

In a third experiment, a positive bias of 0.5 V is injected into the voltage sensor at 400.0 s in q_{11} by spoofing the real sensor data in software. The measured and estimated outputs are shown in Fig. 91. A partial diagnoser is shown in Fig. 92. The symbol generator reports an increase in battery voltage at 400.0 s. The fault candidates generated are $\{(V_B^+, q_{11}), (C_0^-, q_{11})\}$, since no other fault can cause an increase in $V_B(t)$ as the first deviation. At 402.5 s, the change in $V_B(t)$ is determined to not be discrete, but this does not change the current diagnosis. At 407.5 s, the slope is computed to be `0`, which means that either the slope is zero or very small, but the change in $V_B(t)$ is now regarded as a discontinuity. The diagnosis, however, remains $\{(V_B^+, q_{11}), (C_0^-, q_{11})\}$. Because no further measurements deviate, C_0^- cannot be eliminated. If we assume that discontinuities appear at the point of fault occurrence, however, then the fact that the other measurements are still nominal can allow us to reject C_0^- and isolate V_B^+.

We now turn our attention to discrete faults. In the first experiment, we investigate an unexpected switch fault. At 375.5 s, Sw_1 turns off without a command, so the expected mode is q_{11} but the actual mode is $q_{\alpha_0 1}$. The measured and estimated outputs are shown in Fig. 93. The partial diagnoser of Fig. 90 applies to this case also. As a result of the fault, $I_{L1}(t)$ goes imme-

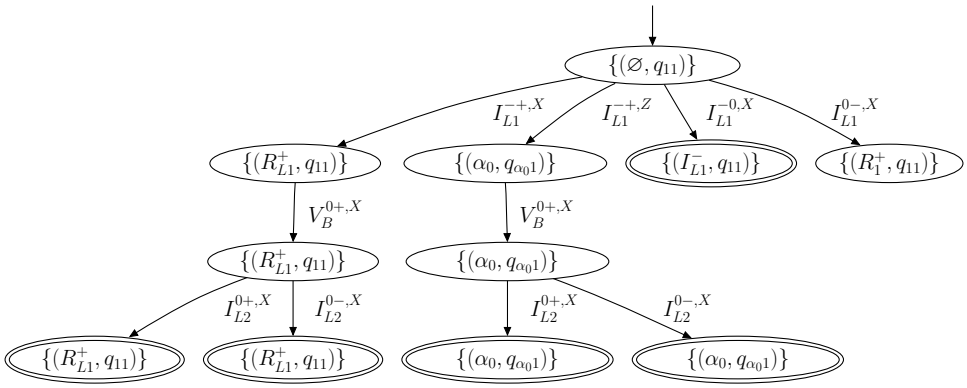


Figure 90: Partial diagnoser for isolating R_{L1}^{+}.

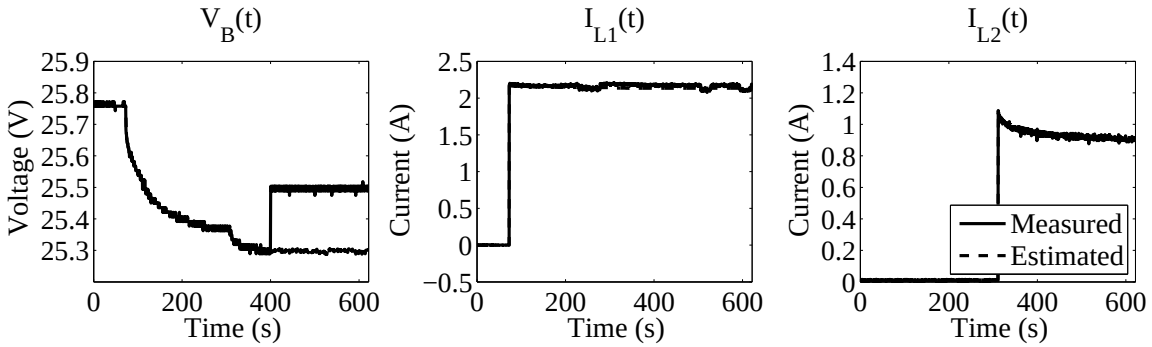


Figure 91: V_B^{+} fault with bias of 0.5.

diately to zero, and $V_B(t)$ increases as a result of less current being drawn. The fault is detected at 376.0 s, and the symbol generator reports a decrease in $I_{L1}(t)$. The initial fault hypotheses are then $\{(R_1^{+}, q_{11}), (R_{L1}^{+}, q_{11}), (I_{L1}^{-}, q_{11}), (\alpha_0, q_{\alpha_0 1})\}$. At 376.5 s, the increase in $V_B(t)$ is detected, so the diagnosis reduces to $\{(R_{L1}^{+}, q_{11}), (\alpha_0, q_{\alpha_0 1})\}$. At 378.5 s, the symbol generator determines that I_{L1}^{-} went to zero, and therefore α_0 is isolated as the true fault.

In the second experiment, we investigate a stuck switch fault. At 414.0 s, Sw_1 is commanded off but remains on. The measured and estimated outputs are shown in Fig. 94. Therefore, the estimated system mode is q_{01} but the actual system mode is $q_{\alpha_1 1}$, and $\hat{I}_{L1}(t)$ goes to zero, while $I_{L1}(t)$ remains nonzero. A partial diagnoser is given in Fig. 95. The fault is detected at 416.0 s, and the symbol generator reports that $I_{L1}(t)$ has increased. Because the expected mode is q_{01}, the only reason for the current to deviate is due to a discrete fault or a sensor fault, so the fault hypotheses are $\{(I_{L1}^{-}, q_{01}), (\alpha_1, q_{\alpha_1 1})\}$. At 418.5 s, the symbol generator reports $I_{L1}(t)$ went to a nonzero value with respect to the estimate of zero. Because we consider only sensor bias, and not failure, α_1 is isolated as the true fault.

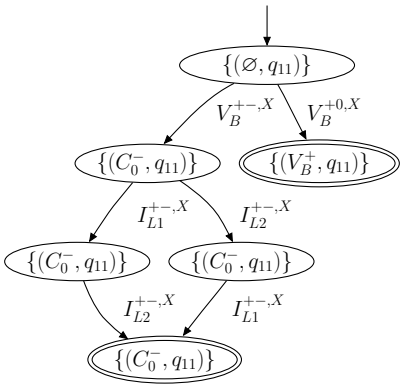


Figure 92: Partial diagnoser for isolating V_B^+.

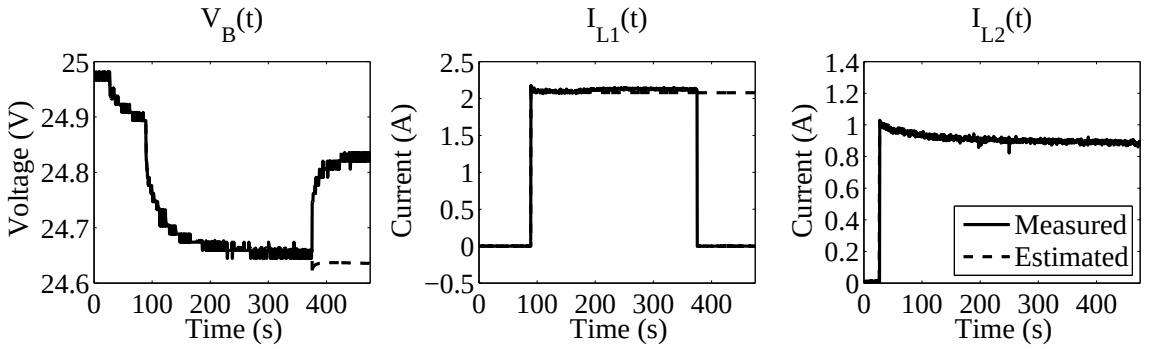


Figure 93: Sw_1 turns off.

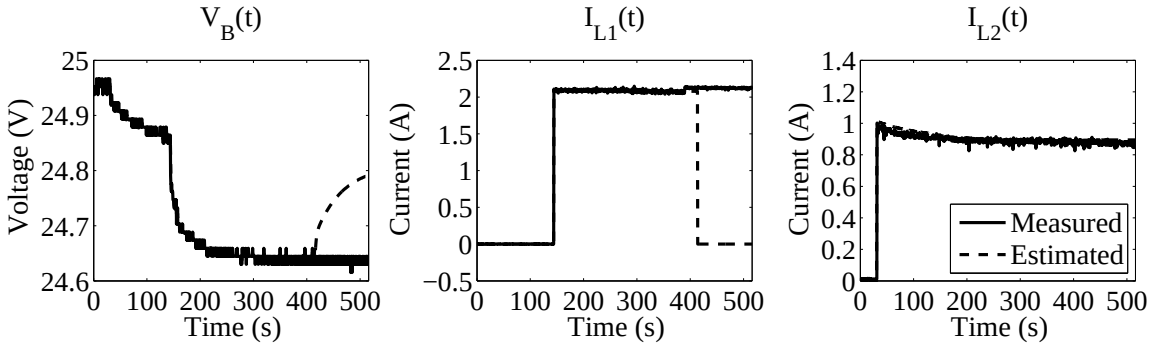


Figure 94: Sw_1 gets stuck on.

In a third experiment, we investigate another unexpected switch fault, where Sw_1 turns on by itself. At 417.5 s, the discrete fault is injected in mode q_{01}, and the actual system mode becomes $q_{\alpha_1 1}$. This fault is the same as in the previous case, only it manifests in the measurements immediately, rather than only when a switching command is issued. The measured and estimated outputs are shown in Fig. 96. A partial diagnoser is given in Fig. 97. This case is treated similarly to the previous case. As a result of the fault, the estimated Load 1 current remains zero, but the measured current

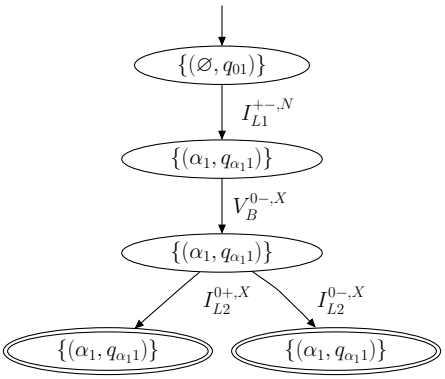


Figure 95: Partial diagnoser for isolating α_1.

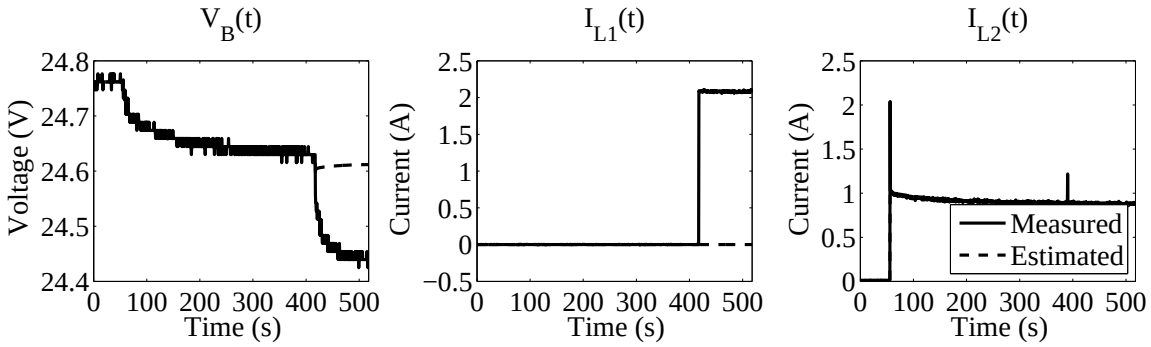


Figure 96: Sw_1 turns on.

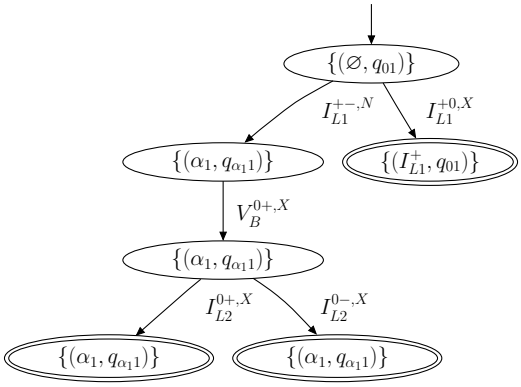


Figure 97: Partial diagnoser for isolating α_1.

goes to a nonzero value. The fault is detected at 417.5 s, and the symbol generator reports that $I_{L1}(t)$ increases with respect to nominal. Because Load 1 is not expected to be on, the diagnosis is only $\{(I_{L1}^{-}, q_{01}), (\alpha_1, q_{\alpha_1 1})\}$. At 419.0 s, the symbol generator reports an increase in $V_B(t)$, thus eliminating the sensor fault as a candidate and isolating α_1 as the fault.

Summary

We presented a case study for hybrid systems diagnosis on the ADAPT system. Theoretical results demonstrated the diagnosability properties of the system. Detailed simulation experiments examined the effects of fault magnitude and sensor noise on the robustness of the approach. Experimental results provided a demonstration of the approach on the ADAPT hardware.

From the results, it is clear that correct symbol generation plays a crucial role in our diagnosis methods. Diagnosers are designed assuming ideal symbol generation, so if this cannot be achieved, then correct candidates may be dropped, or incorrect candidates retained. With improved symbol generation, especially in discontinuity detection, and in computing the slope, which is difficult to tune empirically, diagnosis results will be improved. One way to alleviate the problem is to employ additional sensors that, if present, lead to faster diagnosis, so that additional symbols like the slope may not be necessary to achieve a unique candidate.

The results also demonstrate how controlled mode changes during fault isolation can adversely affect the diagnosis results. From analyzing the diagnoser, we can identify these situations, and use them to block or execute controller actions to avoid them, so that unique candidates can be isolated. Further, they show that the additional symbol introduced for discrete faults, the discrete change feature, is necessary to quickly discriminate discrete from parametric faults, and is very powerful for the type of systems we deal with.

CHAPTER X

CONCLUSIONS

We have presented an approach for multiple fault diagnosis in hybrid systems that extends the TRANSCEND and Hybrid TRANSCEND in many important ways. The approach takes advantage of temporal information in the measurement deviations to enhance fault isolation and systematically construct event-based fault models and diagnosers, which can be used to verify diagnosability of the system and provide particular traces for which ambiguities will arise. We also demonstrated and evaluated the approach on two case studies, a robot formation system and an electrical distribution system.

Summary of Contributions

Our contributions have focused on enhancements to qualitative fault isolation, systematic construction of event-based diagnosers for hybrid systems, and experimental validation of our approach.

1. **Relative Measurement Orderings.** We introduced the notion of *relative measurement orderings*, which define predicted temporal orders of measurement deviations due to faults. We provided an extension of the prediction algorithm in TRANSCEND that computes measurement orderings, based on a sound theory of how they appear in systems. In conjunction with fault signatures, measurement orderings allow us to define event-based *fault models*, where the events are measurement deviations. The fault models are used in the derivation of event-based diagnosers, and form the basis for definitions of distinguishability and diagnosability in our framework.

   Using the case studies, we showed that relative measurement orderings increase the discriminatory ability of the diagnosis algorithms. Therefore, systems which may not have been diagnosable before may now become diagnosable. Further, fewer measurements are needed to deviate before a unique candidate is isolated, so diagnoses are achieved faster. We also showed that relative measurement orderings are critical to correctly defining the behaviors that result from the interactions of multiple faults. If not considered, then it is possible that physically incorrect combinations may be considered during isolation, which decreases the diagnostic precision.

2. **Discrete Faults.** We extended fault isolation from only *parametric faults*, defined by unanticipated changes in a system parameter value, to *discrete faults*, defined by unanticipated changes in the mode of a switching component. This required extensions to the hybrid bond graph modeling language and the temporal causal graph to incorporate discrete faults and their analysis. We extended the hypothesis generation algorithms of TRANSCEND to identify discrete faults as possible causes of faulty behavior, and the prediction algorithm to predict the effects of discrete faults.

   We also enhanced fault signatures to include an additional symbol for a special case of discontinuity that is useful in distinguishing between parametric and discrete faults. Through the case studies, we showed that this symbol is necessary to distinguish between some parametric and discrete faults. We also provided a method to compute this new feature from the estimated and measured outputs.

3. **Multiple Faults.** We extended fault isolation to multiple faults under a best effort paradigm. We showed how fault masking limits the discriminatory ability of the diagnosis algorithms, and the justification for the use of minimal diagnoses within continuous and hybrid systems. We illustrated how the effects of multiple faults can be formed in a consistent manner from the individual faults, for which relative measurement orderings are necessary. We use the principle of minimality and a limit on candidate cardinality in our diagnoses to improve efficiency.

   We also provided simulation results that demonstrated the effect of fault magnitude on the quality of the diagnosis results when multiple faults occurred. This helps to further demonstrate the usefulness of minimal diagnoses and the uncertainty that cannot be avoided using only qualitative information for fault isolation.

4. **Event-based Diagnosability Analysis and Diagnoser Design.** We provided a formal, event-based framework for hybrid systems diagnosis based on the qualitative fault isolation algorithms we presented. Under this framework, we provided algorithms for the systematic construction of event-based diagnosers that track system events and provide associated fault hypotheses in an efficient manner. Because space complexity can be an issue, we also provided a spectrum of implementations and a pruning method which, as shown in the ADAPT case study, can reduce the size of the diagnosers considerably.

   Using our notions of fault traces and languages, we defined distinguishability and diagnosability under our framework. Diagnosability can be automatically verified using the event-based

diagnosers. If the system is not diagnosable, then the diagnosers can also identify which traces lead to ambiguous diagnosis results, and if possible, how these traces can be avoiding by permitting or prohibiting certain controlled mode changes.

5. **Experimental Case Study for Robot Formations.** We provided a case study for robot formations to illustrate our diagnosis approach for single fault diagnosis in continuous systems. The formation study helped to demonstrate the necessity of relative measurement orderings in practical systems, because without them, formation systems are not diagnosable. Because formations are inherently distributed, we applied an extension of the distributed TRANSCEND approach [40, 41] to incorporate orderings.

   We developed bond graph models for mobile robots, and used a well-accepted formation control strategy to link the robots. Other formation control strategies can also be used with our approach, as long as there is a tight coupling between the robots and it allows the propagation of faults between robots to be modeled. Theoretical results verified the scalability of the approach as the number of robots increases, and experimental results validate the approach on an actual robot formation.

6. **Experimental Case Study for Electrical Power Distribution Systems**

   We also provided a case study to the Advanced Diagnostics and Prognostics Testbed. Since ADAPT is modeled as a hybrid system, it best illustrates the advantages of our diagnosis approach. We focused on a particular subset of ADAPT that includes a highly nonlinear battery, two relays, and two DC loads. Therefore both hybrid and nonlinear behaviors are present.

   We analyzed the diagnosability of ADAPT, and provided some partial event-based diagnosers to illustrate the results. We performed extensive diagnosis experiments in simulation in order to study the robustness of our approach with respect to fault magnitude and sensor noise. We also provided some simulations for multiple faults to examine the role of fault masking within ADAPT when multiple faults occur. Experimental results demonstrated our approach on the actual system, and illustrated the importance of relative measurement orderings and the additional qualitative symbol in diagnosing system faults.

Discussion

In Chapter II, we described previous work in model-based diagnosis. As discussed, model-based solutions are best when an appropriate model of the system can be developed. The models can be used as a basis for diagnosis and be used to show analytical properties of the system. In our approach, we use hybrid bond graphs to model the system. From the HBG model, we can automatically derive a hybrid observer and a temporal causal graph. Note that in general, both observers and temporal causal graphs can be generated from other hybrid system models, however, HBGs provide a means for automatic generation. Using the TCG, we can derive fault signatures and measurement orderings, which comprise the diagnostic information of our approach. Signatures and orderings can be derived from other types of models, but there are systematic procedures to generate them from TCGs. Diagnosis uses signatures and orderings within a consistency-based framework to isolate faults. In model-based diagnosis strategies, the observed output is checked for consistency with predicted output in order to derive fault candidates. In our approach, we can efficiently extract from our hybrid system models the information we need for diagnosis, and reasoning using this information is very efficient.

Using signatures and orderings, we can derive event-based fault models, which are then used to systematically create event-based diagnosers. Like discrete-event system approaches, we model faults as unobservable events, and construct offline diagnosers that trace observable events and map them back to fault hypotheses. Essentially, diagnostic information contained within the system model is compiled into a diagnoser. The result is an online diagnoser, which is both time- and space-efficient for continuous systems. For hybrid systems, the compiled diagnoser is still time-efficient, but its space-efficiency is poor and so is useful for offline analysis only. Using our event-based diagnosers, we can define notions of diagnosability that turn out to be similar to those in DES approaches, namely, we must ensure that if a fault occurs, there will be an observable sequence of events that eventually identifies the true fault.

A key advantage to our method is that it is one of the few approaches for hybrid systems diagnosis that can effectively handle both parametric and discrete faults within a common modeling and diagnosis framework. Purely-estimation based approaches can do this, but require that the fault magnitudes are known a priori, which is infeasible in general. We allow parametric faults to be of any magnitude, and fault identification strategies can then be used to determine the magnitude [14,45].

Future Directions

Our approach does have some limitations, and these provide directions for future research. One limitation is that our event-based diagnosers are designed based on ideal symbol generation, i.e., the full signature is known at the point of the first deviation. If symbol generation is not ideal, then the diagnosability analysis of the system may not be correct. We observed this with the ADAPT system, e.g., when a sensor fault occurred we could not differentiate it from a battery capacitance fault, because our symbol generation method cannot verify whether the slope of a residual is truly zero or is just very small, since noise is present. So, even though the two faults are distinguishable according to the diagnoser, we could not distinguish them in practice. Additional temporal information can be used to distinguish further, since the capacitance fault will cause other measurement deviations fairly soon after detection. Such timing information is used in timed DES methods and timed failure propagation graphs [7, 8, 101].

Another problem with nonideal symbol generation arises in multiple fault diagnosis. Since symbol generation takes place over a window of samples, new faults may occur within those windows and affect the results provided by symbol generation, and this may affect fault isolation. True discontinuity detection is also not present, because only deviations in which the computed mean and slope have different symbols are classified as discontinuities. So, discontinuities of the form `++` and `--` are not allowed, however, some sensor faults may produce these signatures, so these signatures should also be considered. Since classification of discontinuities relies on calculation of the slope, the diagnoser can only use the fact that a deviation is a discontinuity after the slope is computed. Therefore, the sooner a discontinuity can be detected, the faster fault isolation will be. Better symbol generation procedures are necessary to compute discontinuities in signals within TRANSCEND based on analysis of the change in the signal at the time of fault detection.

In hybrid systems diagnosis, we assumed only a subset of autonomous mode changes that act as discrete faults. This class of mode changes is sufficient for the ADAPT system, but is not applicable to all systems. With this assumption, we were able to simplify the extension of Hybrid TRANSCEND to an approach which uses a single reference of nominal behavior, and isolating both parametric faults and unpredicted autonomous mode changes based on the deviations they produce from that reference. As future work, we can apply our methods to the multiple-reference approach of Hybrid TRANSCEND, which will also allow us to deal with a richer class of autonomous mode changes, such as those that cause state resets.

Another limitation is that we assume parametric faults have abrupt profiles. Faults may also be incipient or have arbitrary profiles. Also, both parametric and discrete faults may be intermittent, whereas we assumed all faults are persistent. In order to deal with these types of fault profiles, a tighter integration with quantitative methods is necessary, because fault isolation based on qualitative information alone becomes limited. For example, qualitative fault isolation for multiple faults can only be a best effort paradigm, because fault masking means that effects due to masked faults may be missed. In estimation-based diagnosis approaches for hybrid systems, fault isolation is performed using only quantitative information. The problem with these approaches, however, is that there are many possible modes to track, and fault modes have low probability so are typically missed. A recent trend has been to incorporate this style of diagnosis with reasoners (e.g., [14]) in order to avoid these problems and focus estimation only on those modes consistent with the observed faulty behavior. Recent work using TRANSCEND in this type of framework has been presented in [78], and the contributions of this dissertation made to qualitative fault isolation in the TRANSCEND framework makes TRANSCEND a better reasoner for this type of integrated diagnosis approach. This approach integrates well with the multiple-reference approach used for hybrid diagnosis by Hybrid TRANSCEND.

In hybrid systems, quantitative methods may need to keep track of multiple possible modes when a fault is detected. In our framework, we assume that we have the correct system mode up to the point of fault occurrence, after which unpredicted autonomous mode changes, such as discrete faults, may occur, along with controlled mode changes. We perform diagnosis over measurement deviations with respect to a particular reference mode, determined by issuing only the controlled mode change commands. Therefore, each candidate represents which parametric faults and which unpredicted mode changes may have occurred with respect to the reference mode. The approach can be extended by having multiple references, each made based on hypothesized mode changes induced by discrete faults or other unpredicted autonomous mode changes, and performing qualitative diagnosis in each mode. This would also enable the approach to take advantage of sensors which monitor the states of autonomously switching components, such as circuit breakers. Assuming the sensor is correct produces one possible mode hypothesis in which the mode has changed, and assuming it is faulty produces another in which the mode has not changed. These extensions should also be more tightly integrated with quantitative methods to ensure that modes that do not track well can be dropped from consideration.

Fault identification has been presented for the TRANSCEND framework previously for single, abrupt, parametric faults in continuous and hybrid systems [14, 45]. The approach is based on

parameter estimation based on the isolated fault. The parameter estimation simulates the system behavior for different fault magnitudes after fault detection and matches it to the observed faulty behavior. Error signals are generated for selected parameter values to converge onto correct parameter values. Accounting for discrete faults in this approach is easy, because only a single simulation where the discrete fault is applied is needed to generate an error signal. Identifying multiple faults using this approach does not work, however, because the fault identification operates only after fault isolation has completed, and the times of fault occurrence for the different faults are not known. If fault identification becomes merged with isolation, then both aspects become improved by the other. Fault isolation generates only consistent candidates, and fault identification provides the diagnoser with which candidates are unlikely and can be dropped. The diagnosis problem becomes an estimation problem, but can be improved with the use of a reasoner.

APPENDIX A

FACT: FAULT ADAPTIVE CONTROL TECHNOLOGY

Most of the work presented in this dissertation has been implemented as part of the Fault Adaptive Control Technology (FACT) tool suite [158]. FACT consists of three main parts, (*i*) a modeling paradigm, (*ii*) model translators, and (*iii*) runtime diagnosers. Diagnosis in FACT uses both temporal causal graphs and timed failure propagation graphs (TFPGs) [135]. We omit discussion of the TFPG aspects of FACT.

Modeling Paradigm

FACT is based on principles of model-integrated computing (MIC), and utilizes the Generic Modeling Environment [159]. The modeling paradigm of FACT allows for graphical modeling of component-based hybrid systems using hybrid bond graphs [117]. The modeling environment contains bond graph elements and interface elements. Currently, the language assumes that CSPECs have two nominal states, one for the on mode, and one for the off mode of a junction, and two nominal transitions going from on to off. Nominal transitions between the states are specified by guards that are functions of system variables and inputs. To model discrete faults, we allow junctions to be marked as fault candidates. Within the diagnoser, these are then mapped back to common inputs controlling the junctions. Discrete faults are associated with these inputs and stuck-on and stuck-off modes assumed for the controlled junctions. We have also extended the modeling paradigm to allow sensor components to be marked as fault candidates, and the diagnoser assumes abrupt profiles for the sensor faults.

Model Translators

Using the principles of MIC, model translators, or interpreters, are available that transform the HBG model constructed in the modeling framework into some useful form. One of these forms is a flat model representation. The corresponding interpreter has been modified to include the additional information required for discrete faults and sensor faults. A second interpreter creates simulation models of the HBGs for Matlab Simulink® [160], and has been presented in [108, 109, 116]. The simulations allow for simulation of both nominal and faulty data. The interpreter has been extended to build into the simulations injection of discrete faults and sensor faults.

Diagnosers

The diagnoser implements the TRANSCEND and Hybrid TRANSCEND algorithms for system tracking, fault detection, symbol generation, fault isolation, and fault identification. Symbol generation has been extended to include the additional symbol used for distinguishing discrete and parametric faults, as described in Chapter III. The TCGs have been updated as described in Chapter IV to include discrete faults and generic sensor models, for sensor faults with abrupt profiles. Fault isolation has been updated by including isolation of sensor faults and discrete faults, and by including relative measurement orderings, based on the algorithms presented in Chapter V and VI. In addition, the procedures described for constructing event-based diagnosers in Chapter VII have been implemented and were used to automatically generate the diagnoser figures for this dissertation.

REFERENCES

[1] W. Hamscher, L. Console, and J. de Kleer, Eds., *Readings in Model-Based Diagnosis*. Morgan Kaufmann, 1992.

[2] R. Reiter, "A theory of diagnosis from first principles," in *Readings in Nonmonotonic Reasoning*, M. L. Ginsberg, Ed. Los Altos, California: Morgan Kaufmann, 1987, pp. 352–371.

[3] J. de Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, pp. 97–130, 1987.

[4] B. C. Williams and P. P. Nayak, "A model-based approach to reactive self-configuring systems," in *Proceedings of AAAI-96*. AAAI Press, 1996, pp. 971–978.

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

[6] S. H. Zad, R. Kwong, and W. Wonham, "Fault diagnosis in discrete-event systems: framework and model reduction," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1199–1212, July 2003.

[7] Y.-L. Chen and G. Provan, "Modeling and diagnosis of timed discrete event systems – a factory automation example," in *Proceedings of the American Control Conference*, June 1997, pp. 31–36.

[8] S. H. Zad, R. H. Kwong, and W. M. Wonham, "Fault diagnosis in timed discrete-event systems," in *Proceedings of the 38th IEEE Conference on Decision and Control*, Dec. 1999, pp. 1756–1761.

[9] J. Gertler, "Fault detection and isolation using parity relations," *Control Engineering Practice*, vol. 5, no. 5, pp. 653–661, 1997.

[10] ——, *Fault Detection and Diagnosis in Engineering Systems*. New York: Marcel Dekker, 1998.

[11] R. J. Patton and J. Chen, "Observer-based fault detection and isolation: robustness and applications," *Control Engineering Practice*, vol. 5, no. 5, pp. 671–682, 1997.

[12] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy — a survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[13] R. Isermann, "Supervision, fault-detection, and fault-diagnosis methods — an introduction," *Control Engineering Practice*, vol. 5, no. 5, pp. 639–652, 1997.

[14] S. Narasimhan and G. Biswas, "Model-based diagnosis of hybrid systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 37, no. 3, pp. 348–361, May 2007.

[15] M. Hofbaur and B. Williams, "Hybrid diagnosis with unknown behavioral modes," in *Proceedings of the 13th International Workshop on Principles of Diagnosis*, May 2002, pp. 97–105.

[16] X. Koutsoukos, J. Kurien, and F. Zhao, *Hybrid Systems: Computation and Control*, ser. LNCS. Springer, 2003, vol. 2623, ch. Estimation of Distributed Hybrid Systems Using Particle Filtering Methods, pp. 298–313.

[17] S. McIlraith, "Diagnosing hybrid systems: a bayesian model selection approach," in *Proceedings of the 11th International Workshop on Principles of Diagnosis*, 2000, pp. 140–146.

[18] S. Narasimhan and L. Brownston, "HyDE — a general framework for stochastic and hybrid model-based diagnosis," in *Proc. of the 18th Int. Workshop on Principles of Diagnosis*, May 2007, pp. 162–169.

[19] F. Zhao, X. D. Koutsoukos, H. W. Haussecker, J. Reich, and P. Cheung, "Monitoring and fault diagnosis of hybrid systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 6, pp. 1225–1240, 2005.

[20] P. Mosterman and G. Biswas, "Diagnosis of continuous valued systems in transient operating regions," *IEEE Trans. SMC, Part A*, vol. 29, no. 6, pp. 554–565, 1999.

[21] P. J. Mosterman, "Hybrid dynamic systems: A hybrid bond graph modeling paradigm and its application in diagnosis." Ph.D. dissertation, Vanderbilt University, 1997.

[22] S. Narasimhan, "Model-based diagnosis of hybrid systems," Ph.D. dissertation, Vanderbilt University, 2002.

[23] J. M. Kościelny and K. Zakroczymski, "Fault isolation method based on time sequences of symptom appearance," in *Proceedings of IFAC SafeProcess 2000*, 2000.

[24] J. M. Kościelny, "Fault isolation in industrial processes by the dynamic table of states method," *Automatica*, vol. 31, no. 5, pp. 747–753, 1995.

[25] V. Puig, J. Quevedo, T. Escobet, and B. Pulido, "On the integration of fault detection and isolation in model-based fault diagnosis," in *Proceedings of the 16th International Workshop on Principles of Diagnosis (DX-05)*, 2005, pp. 227–232.

[26] V. Puig, F. Schmid, J. Quevedo, and B. Pulido, "A new fault diagnosis algorithm that improves the integration of fault detection and isolation," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Dec. 2005, pp. 3809–3814.

[27] M. Daigle, X. Koutsoukos, and G. Biswas, "Relative measurement orderings in diagnosis of distributed physical systems," in *43rd Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2005, pp. 1707–1716.

[28] M. J. Daigle, X. D. Koutsoukos, and G. Biswas, "Distributed diagnosis in formations of mobile robots," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 353–369, Apr. 2007.

[29] S. A. McIlraith, G. Biswas, D. Clancy, and V. Gupta, *Hybrid Systems: Computation and Control*, ser. LNCS. Springer, 2000, vol. 1790, ch. Hybrid Systems Diagnosis, pp. 282–295.

[30] M. Hofbaur and B. Williams, "Mode estimation of probabilistic hybrid systems," in *Fifth International Workshop on Hybrid Systems: Computation and Control*. Springer, 2002, pp. 253–266.

[31] R. Dearden and D. Clancy, "Particle filters for real-time fault detection in planetary rovers," in *Proceedings of the 12th International Workshop on Principles of Diagnosis*, 2001, pp. 1–6.

[32] S. Narasimhan, R. Dearden, and E. Benazera, "Combining particle filters and consistency-based approaches for monitoring and diagnosis of stochastic hybrid systems," in *Proceedings of the 15th International Workshop on Principles of Diagnosis*, June 2004.

[33] X. Koutsoukos, J. Kurien, and F. Zhao, "Monitoring and diagnosis of hybrid systems using particle filtering methods," in *Proceedings 15th Int. Symposium on Mathematical Theory Networks and Systems*, 2002.

[34] W. Wang, L. Li, D. Zhou, and K. Liu, "Robust state estimation and fault diagnosis for uncertain hybrid nonlinear systems," *Nonlinear Analysis: Hybrid Systems*, vol. 1, no. 1, pp. 2–15, Mar. 2007.

[35] V. Cocquempot, T. El Mezyani, and M. Staroswiecki, "Fault detection and isolation for hybrid systems using structured parity residuals," in *Proceedings of the 5th Asian Control Conference*, 2004, pp. 1204–1212.

[36] M. Daigle, X. Koutsoukos, and G. Biswas, "An integrated approach to parametric and discrete fault diagnosis in hybrid systems," in *HSCC 2008*, ser. LNCS. Springer-Verlag, 2008, vol. 4981, pp. 614–617.

[37] ——, "An integrated approach to parametric and discrete fault diagnosis in hybrid systems," Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA, Tech. Rep. ISIS-07-815, Jan. 2008.

[38] ——, "Multiple fault diagnosis in complex physical systems," in *Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06)*, June 2006, pp. 69–76.

[39] ——, "A qualitative approach to multiple fault isolation in continuous systems," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, July 2007, pp. 293–298.

[40] I. Roychoudhury, G. Biswas, X. Koutsoukos, and S. Abdelwahed, "Designing distributed diagnosers for complex physical systems," in *Proceedings of the 16th International Workshop on Principles of Diagnosis (DX 05)*, Monterey, California, June 2005, pp. 31–36.

[41] I. Roychoudhury, G. Biswas, and X. Koutsoukos, "Designing distributed diagnosers for complex continuous systems," *IEEE Transactions on Automation Science and Engineering*, 2008.

[42] S. Poll, A. Patterson-Hine, J. Camisa, D. Garcia, D. Hall, C. Lee, O. Mengshoel, C. Neukom, D. Nishikawa, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos, "Advanced diagnostics and prognostics testbed," in *18th International Workshop on Principles of Diagnosis*, to appear.

[43] S. Poll, A. Patterson-Hine, J. Camisa, D. Nishikawa, L. Spirkovska, D. Garcia, D. Hall, C. Neukom, A. Sweet, S. Yentus, C. Lee, J. Ossenfort, I. Roychoudhury, M. Daigle, G. Biswas, X. Koutsoukos, and R. Lutz, "Evaluation, selection, and application of model-based diagnosis tools and approaches," in *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, May 2007.

[44] G. Biswas, G. Simon, N. Mahadevan, S. Narasimhan, J. Ramirez, and G. Karsai, "A robust method for hybrid diagnosis of complex systems," in *Proceedings of the 5th Symposium on Fault Detection, Supervision and Safety for Technical Processes*, June 2003, pp. 1125–1131.

[45] E.-J. Manders, S. Narasimhan, G. Biswas, and P. Mosterman, "A combined qualitative/quantitative approach for fault isolation in continuous dynamic systems," in *SafeProcess 2000*, vol. 1, Budapest, Hungary, June 2000, pp. 1074–1079.

[46] E.-J. Manders and G. Biswas, "FDI of abrupt faults with combined statistical detection and estimation and qualitative fault isolation," in *Proceedings of the 5th Symposium on Fault Detection, Supervision and Safety for Technical Processes*, June 2003, pp. 347–352.

[47] E.-J. Manders, P. Mosterman, and G. Biswas, "Signal to symbol transformation techniques for robust diagnosis in TRANSCEND," in *Proceedings of the 10th International Workshop on Principles of Diagnosis*, 1999, pp. 155–165.

[48] P. J. Mosterman and G. Biswas, "A theory of discontinuities in physical system models," *Journal of the Franklin Institute*, vol. 335B, no. 3, pp. 401–439, Jan. 1998.

[49] S. Narasimhan and G. Biswas, "An approach to model-based diagnosis of hybrid systems," in *HSCC 2002*, ser. LNCS, C. Tomlin and M. Greenstreet, Eds. Springer-Verlag, 2007, vol. 2289, pp. 308–322.

[50] S. Narasimhan, G. Biswas, G. Karsai, T. Pasternak, and F. Zhao, "Building observers to address fault isolation and control problems in hybrid dynamic systems," in *Proceedings of 2000 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, 2000, pp. 2393–2398.

[51] S. Narasimhan, P. J. Mosterman, and G. Biswas, "A systematic analysis of measurement selection algorithms for fault isolation in dynamic systems," in *Proc. of DX 1998*, Cape Cod, MA USA, May 1998, pp. 94–101.

[52] R. Isermann, "Model-based fault detection and diagnosis – status and applications," in *Proceedings of the 16th IFAC Symposium on Automatic Control in Aerospace*, June 2004, pp. 43–54.

[53] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis Part I: Quantitative model-based methods," *Computers and Chemical Engineering*, vol. 27, pp. 293–311, 2003.

[54] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, "A review of process fault detection and diagnosis Part II: Qualitative models and search strategies," *Computers and Chemical Engineering*, vol. 27, pp. 313–326, 2003.

[55] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis Part III: Process history based methods," *Computers and Chemical Engineering*, vol. 27, pp. 327–346, 2003.

[56] J. de Kleer and B. C. Williams, "Diagnosis with behavioral modes," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Detroit, MI, USA: Morgan Kaufmann, Aug. 1989, pp. 1324–1330.

[57] J. de Kleer, A. K. Mackworth, and R. Reiter, "Characterizing diagnoses and systems," *Artificial Intelligence*, vol. 56, no. 2–3, pp. 197–222, Aug. 1992.

[58] M. Cordier and S. Thiébaux, "Event-based diagnosis for evolutive systems," IRISA, Cedex, France, Tech. Rep. 819, 1994.

[59] S. McIlraith, "Explanatory diagnosis: Conjecturing actions to explain observations," in *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR98)*, 1998, pp. 167–177.

[60] V. Brusoni, L. Console, P. Terenziani, , and D. T. Dupré, "Characterizing temporal abductive diagnosis," in *Proceedings of the 6th International Workshop on Principles of Diagnosis*, 1995.

[61] V. Brusoni, L. Console, P. Terenziani, and D. T. Dupré, "An efficient algorithm for computing temporal abductive diagnoses," in *Proceedings of the 6th International Workshop on Principles of Diagnosis*, 1995.

[62] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, Sept. 1995.

[63] S. Jiang and R. Kumar, "Failure diagnosis of discrete-event systems with linear-time temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 49, no. 6, pp. 934–945, June 2004.

[64] S. Tripakis, "Fault diagnosis for timed automata," in *Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT02)*, ser. Lecture Notes in Computer Science, vol. 2469. Springer, 2002, pp. 205–221.

[65] S. Bibas, M.-O. Cordier, P. Dague, C. Dousson, F. Lvy, and L. Roz, "Alarm driven supervision for telecommunication network: I – off-line scenarios generation," *Annales des Telecommunications*, vol. 51, no. 910, pp. 493–500, 1996.

[66] M.-O. Cordier and C. Dousson, "Alarm driven monitoring based on chronicles," in *Proceedings of the 4th Symposium on Fault Detection Supervision and Safety for Technical Processes (Safeprocess 2000)*, June 2000, pp. 286–291.

[67] C. Dousson, "Alarm driven supervision for telecommunication network: Ii – on-line chronicle recognition," *Annales des Telecommunications*, vol. 51, no. 910, pp. 501–508, 1996.

[68] J. Lunze, "Diagnosis of quantized systems based on a timed discrete-event model," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 30, no. 3, pp. 322–335, 2000.

[69] J. Lunze and J. Shröder, "Sensor and actuator fault diagnosis of systems with discrete inputs and outputs," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 2, pp. 1096–1107, Apr. 2004.

[70] R. Isermann and P. Ballé, "Trends in the application of model-based fault detection and diagnosis of technical processes," *Control Engineering Practice*, vol. 5, no. 5, pp. 709–719, 1997.

[71] O. Dressler, "Model-based diagnosis on board: Magellan-mt inside," in *Proceedings of the Fifth International Workshop on Principles of Diagnosis*, 1994, pp. 87–92.

[72] F. Lackinger and W. Nejdl, "Integrating model-based monitoring and diagnosis of complex dynamic systems," in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1991, pp. 1123–1128.

[73] H. T. Ng, "Model-based, multiple fault diagnosis of time-varying, continuous physical devices," in *Sixth Conference on Artificial Intelligence Applications*, vol. 1, May 1990, pp. 9–15.

[74] S. Subramanian and R. J. Mooney, "Qualitative multiple-fault diagnosis of continuous dynamic systems using behavioral modes," in *The 1996 13th National Conference on Artificial Intelligence*, Aug. 1996, pp. 965–970.

[75] E. Benazera, L. Travé-Massuyès, and P. Dague, "State tracking of uncertain hybrid concurrent systems," in *Proc. of the 13th Int. Workshop on Principles of Diagnosis*, 2002, pp. 106–114.

[76] G. K. Fourlas, K. J. Kyriakopoulos, and N. J. Krikelis, "Fault diagnosis of hybrid systems," in *Proc. of the 2005 IEEE Int. Symp. on Intelligent Control*, June 2005, pp. 832–837.

[77] M. D. Di Benedetto, S. Di Gennaro, and A. D'Innocenzo, "Diagnosability verification for hybrid automata," in *Hybrid Systems: Computation and Control*, ser. LNCS. Springer, 2007, vol. 4416, pp. 684–687.

[78] I. Roychoudhury, G. Biswas, and X. Koutsoukos, "A Bayesian approach to efficient diagnosis of incipient faults," in *Proceedings of the 17th International Workshop on Principles of Diagnosis (DX 06)*, June 2006, pp. 243–250.

[79] U. Lerner, R. Parr, D. Koller, and G. Biswas, "Bayesian fault detection and diagnosis in dynamic systems," in *Proc. of Seventeenth National Conference on Artificial Intelligence*, 2000, pp. 531–537.

[80] M. R. Garey and D. S. Johnson, *Computers and intractability; a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[81] V. Brusoni, L. Console, P. Terenziani, and D. T. Dupr, "A spectrum of definitions for temporal model-based diagnosis," *Artificial Intelligence*, vol. 102, no. 1, pp. 39–80, 1998.

[82] M. Daigle, X. Koutsoukos, and G. Biswas, "A discrete event approach to diagnosis of continuous systems," in *Proceedings of the 18th International Workshop on Principles of Diagnosis*, May 2007, pp. 259–266.

[83] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[84] B. A. Brandin and W. M. Wonham, "Supervisory control of timed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 329–342, Feb. 1994.

[85] X. Koutsoukos, P. Antsaklis, J. Stiver, and M. Lemmon, "Supervisory control of hybrid systems," *Proceedings of IEEE*, vol. 88, no. 7, pp. 1026–1049, July 2000.

[86] M. Daigle, X. Koutsoukos, and G. Biswas, "Fault diagnosis of continuous systems using discrete-event methods," in *Proceedings of the 46th IEEE Conference on Decision and Control*, Dec. 2007, pp. 2626–2632.

[87] B. Kuipers, "Qualitative simulation," *Artificial Intelligence*, vol. 29, pp. 289–338, 1986.

[88] P. Struss, A. Malik, and M. Sachenbacher, "Qualitative modeling is the key to automated diagnosis," in *Proceedings of the 13th IFAC World Congress*. Pergamon, 1996.

[89] A. Malik and P. Struss, "Diagnosis of dynamic systems does not necessarily require simulation," in *Proceedings of DX97*, 1997, pp. 147–156.

[90] F. Cascio, L. Console, M. Guagliumi, M. Osella, S. Sottano, and D. Theseider, "Generating on-board diagnostics of dynamic automotive systems based on qualitative deviations," *AI Communications*, vol. 12, no. 1, pp. 33–44, 1999.

[91] L. Console, C. Picardi, and M. Ribaudo, "Process algebras for systems diagnosis," *Artificial Intelligence*, vol. 142, no. 1, pp. 19–51, 2002.

[92] M.-O. Cordier, P. Dague, F. Lévy, J. Montmain, M. Staroswiecki, and L. Travé-Massuyès, "Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 5, pp. 2163–2177, 2004.

[93] B. Halder and N. Sarkar, "Robust fault detection based on nonlinear analytic redundancy techniques with application to robotics," in *Proceedings of IMECE*, Nov. 2005, pp. 2864–2869.

[94] S. Diop, "Elimination in control theory," *Mathematics of Control, Signals, and Systems*, vol. 4, no. 1, pp. 17–32, Mar. 1991.

[95] M. Daigle, X. Koutsoukos, and G. Biswas, "Distributed diagnosis of coupled mobile robots," in *Proceedings 2006 IEEE International Conference on Robotics and Automation*, May 2006, pp. 3787–3794.

[96] ——, "On discrete event diagnosis methods for continuous systems," in *Proceedings of the IEEE 15th Mediterranean Conference on Control and Automation*, June 2007.

[97] C. Goodrich and J. Kurien, "Continuous measurements and quantitative constraints — challenge problems for discrete modeling techniques," in *Proc. of iSAIRAS-2001*, 2001.

[98] T. A. Henzinger, "The theory of hybrid automata," in *Proceedings of the Eleventh Annual IEEE Symposium On Logic In Computer Science (LICS'96)*. New York, USA: IEEE Computer Society Press, July 1996, pp. 278–293.

[99] X. Koutsoukos, F. Zhao, H. Haussecker, J. Reich, and P. Cheung, "Fault modeling for monitoring and diagnosis of sensor-rich hybrid systems," in *Proceedings of the 40th IEEE Conference on Decision and Control*, Dec. 2001, pp. 793–801.

[100] J. de Kleer, "Diagnosing intermittent faults," in *Proceedings of the 18th International Workshop on Principles of Diagnosis*, May 2007, pp. 45–51.

[101] S. Abdelwahed and G. Karsai, "Practical considerations in systems diagnosis using timed failure propagation graph models," in *Proceedings of Autotestcon 2007*, Sept. 2007, pp. 129–136.

[102] O. Contant, S. Lafortune, , and D. Teneketzis, "Failure diagnosis of discrete event systems: The case of intermittent faults," in *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 4, 2002, pp. 4006–4011.

[103] A. Correcher, E. Garcia, F. Morant, E. Quiles, and E. Blasco-Gimenez, "Intermittent failure diagnosis in industrial processes," in *IEEE International Symposium on Industrial Electronics*, vol. 2, 2003, pp. 723– 728.

[104] S. Soldani, M. Combacau, A. Subias, and J. Thomas, "Intermittent fault diagnosis: a diagnoser derived from the normal behavior," in *Proceedings of the 18th International Workshop on Principles of Diagnosis*, May 2007, pp. 391–398.

[105] H. Chung, Z. Bien, J. Park, and P. Seong, "Incipient multiple fault diagnosis in real time with application to large-scale systems," *IEEE Transactions on Nuclear Science*, vol. 41, no. 4, pp. 1692–1703, Aug. 1994.

[106] X. Zhang, M. M. Polycarpou, and T. Parisini, "A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 4, pp. 576 – 593, Apr. 2002.

[107] M. A. Demetriou and M. M. Polycarpou, "Incipient fault diagnosis of dynamical systems using on-line approximators," *IEEE Transactions on Automatic Control*, vol. 43, no. 11, pp. 1692–1617, Nov. 1998.

[108] I. Roychoudhury, M. Daigle, G. Biswas, X. Koutsoukos, , and P. J. Mosterman, "A method for efficient simulation of hybrid bond graphs," in *International Conference on Bond Graph Modeling and Simulation (ICBGM 2007)*, Jan. 2007, pp. 177–184.

[109] M. Daigle, I. Roychoudhury, G. Biswas, and X. Koutsoukos, "Efficient simulation of component-based hybrid models represented as hybrid bond graphs," in *HSCC 2007*, ser. LNCS, A. Bemporad, A. Bicchi, and G. Butazzo, Eds. Springer-Verlag, 2007, vol. 4416, pp. 680–683.

[110] R. E. Kirk, *Statistics: An Introduction*. Fort Worth: Harcourt Brace, 1999.

[111] M. Basseville and I. Nikiforov, *Detection of Abrupt Changes - Theory and Application*. Prentice-Hall Inc., 1993.

[112] E.-J. Manders, "A combined statistical detection and qualitative fault isolation scheme for abrupt faults in dynamic systems," Ph.D. dissertation, Vanderbilt University, 2003.

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

[114] B. O. Bouamama, K. Medjaher, M. Bayart, A. K. Samantaray, and B. Conrard, "Fault detection and isolation of smart actuators using bond graphs and external models," *Control Engineering Practice*, no. 2, pp. 159–175, Feb. 2005.

[115] A. Samantaray, K. Medjaher, B. O. Bouamama, M. Staroswiecki, and G. Dauphin-Tanguy, "Diagnostic bond graphs for online fault detection and isolation," *Simulation Modelling Practice and Theory*, no. 3, pp. 237–262, Apr. 2006.

[116] M. Daigle, I. Roychoudhury, G. Biswas, and X. Koutsoukos, "Efficient simulation of component-based hybrid models represented as hybrid bond graphs," Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA, Tech. Rep. ISIS-06-712, Dec. 2006.

[117] E.-J. Manders, G. Biswas, N. Mahadevan, and G. Karsai, "Component-oriented modeling of hybrid dynamic systems using the Generic Modeling Environment," in *Proceedings of the 4th Workshop on Model-Based Development of Computer Based Systems*, Mar. 2006, pp. 159–168.

[118] J. Granda, "The role of bond graph modeling and simulation in mechatronic systems, and integrated software tool: Camp-g, matlab-simulink," *Mechatronics*, vol. 12, pp. 1271–1295, 2002.

[119] K. Edström, "Simulation of mode switching systems using switched bond graphs," Ph.D. dissertation, Linköpings Universitet, Dec. 1996.

[120] J. F. Broenink and K. C. Wijbrans, "Describing discontinuities in bond graphs," in *Proceedings of the International Conference on Bond Graph Modeling*, San Diego, California, 1993, pp. 120–125.

[121] J. Buisson, H. Cormerais, and P.-Y. Richard, "Analysis of the bond graph model of hybrid physical systems with ideal switches," *Proc Instn Mech Engrs Vol 216 Part I: J Systems and Control Engineering*, pp. 47–63, 2002.

[122] H. Vedam and H. Venkatasubramanian, "Signed digraph-based multiple fault diagnosis," *Computers and Chemical Engineering*, vol. 21, pp. 665–660, 1997.

[123] G. Lee, B. Lee, E. Yoon, and C. Han, "Multiple-fault diagnosis under uncertain conditions by the quantification of qualitative relations," *Industrial & Engineering Chemistry Research*, vol. 38, pp. 988–998, 1999.

[124] S. Chessa and P. Santi, "Operative diagnosis of graph-based systems with multiple faults," *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 31, no. 2, pp. 112–119, 2001.

[125] F. Tu, K. R. Pattipati, S. Deb, and V. N. Malepati, "Computationally efficient algorithms for multiple fault diagnosis in large graph-based systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 33, no. 1, pp. 73–85, 2003.

[126] S. Singh, S. Ruan, K. Choi, K. Pattipati, P. Willett, S. M. Namburu, S. Chigusa, D. V. Prokhorov, and L. Qiao, "An optimization-based method for dynamic multiple fault diagnosis problem," in *Proceedings of the 2007 IEEE Aerospace Conference*, March 2007.

[127] S. Singh, K. Choi, A. Kodali, K. Pattipati, J. Sheppard, S. M. Namburu, S. Chigusa, D. V. Prokhorov, and L. Qiao, "Dynamic multiple fault diagnosis problem formulations and solution techniques," in *Proceedings of the 18th International Workshop on Principles of Diagnosis*, May 2007, pp. 383–390.

[128] M. Shakeri, V. Raghavan, K. Pattipati, and A. Patterson-Hine, "Sequential testing algorithms for multiple fault diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 30, no. 1, Jan. 2000.

[129] Z. Wang, M. Marek-Sadowska, K. Tsai, and J. Rajski, "Multiple fault diagnosis using n-detection tests," in *Proceedings of the 21st International Conference on Computer Design (ICCD'03)*, 2003, pp. 198–201.

[130] J. Kurien, X. Koutsoukos, and F. Zhao, "Distributed diagnosis of networked embedded systems," in *Proceedings of the 13th International Workshop on Principles of Diagnosis (DX-2002)*, Semmering, Austria, May 2002, pp. 179–188.

[131] A. Benveniste, E. Fabre, S. Haar, and C. Jard, "Diagnosis of asynchronous discrete-event systems: a net unfolding approach," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 714–727, May 2003.

[132] V. Chandra, Z. Huang, and R. Kumar, "Automated control synthesis for an assembly line using discrete event system control theory," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 33, no. 2, pp. 284–289, May 2003.

[133] M. J. Daigle, X. Koutsoukos, and G. Biswas, "A qualitative event-based approach to continuous systems diagnosis," *IEEE Transactions on Control Systems Technology*, in review.

[134] L. Travé-Massuyès, T. Escobet, and X. Olive, "Diagnosability analysis based on component-supported analytical redundancy relations," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 36, no. 6, pp. 1146–1160, Nov. 2006.

[135] S. Abdelwahed, G. Karsai, and G. Biswas, "Consistency-based robust diagnosis approach for temporal causal systems," in *Proceedings of the 16th International Workshop on Principles of Diagnosis*, June 2005, pp. 73–79.

[136] S. Abdelwahed and G. Karsai, "Notions of diagnosability for timed failure propagation graphs," in *IEEE Systems Readiness Technology Conference*, Sept. 2006, pp. 643–648.

[137] M.-O. Cordier, L. Travé-Massuyès, and X. Pucel, "Comparing diagnosability in continuous and discrete-event systems," in *Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06)*, 2006, pp. 55–60.

[138] M. Bayoudh, L. Traveé-Massuyès, and X. Olive, "Hybrid systems diagnosability by abstracting faulty continuous dynamics," in *Proceedings of the 17th International Principles of Diagnosis Workshop*, 2006, pp. 9–15.

[139] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, Dec 2001.

[140] ——, "Controlling formations of multiple mobile robots," in *Proceedings 1998 IEEE International Conference on Robotics and Automation*, 1998, pp. 2864–2869.

[141] T. Balch and R. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, Dec. 1998.

[142] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[143] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, Apr. 1998.

[144] Z. Duan, Zi-xing Cai, and J. Yu, "Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 3428–3433.

[145] M. Ji, Z. Zhang, G. Biswas, and N. Sarkar, "Hybrid fault adaptive control of a wheeled mobile robot," *IEEE/ASME Transactions on Mechatronics*, vol. 8, no. 2, pp. 226–233, June 2003.

[146] V. Verma, G. Gordon, R. Simmons, and S. Thrun, "Real-time fault diagnosis," *IEEE Robotics & Automation Magazine*, vol. 11, no. 2, pp. 56–66, June 2004.

[147] W. Dixon, I. Walker, and D. Dawson, "Fault detection for wheeled mobile robots with parametric uncertainty," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, vol. 2, July 2001, pp. 1245–1250.

[148] S. Roumeliotis, G. Sukhatme, and G. Bekey, "Sensor fault detection and identification in a mobile robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, Victoria, BC, Canada, Oct 1998, pp. 1383–1388.

[149] P. Goel, G. Dedeoglu, S. Roumeliotis, and G. Sukhatme, "Fault detection and identification in a mobile robot using multiple model estimation and neural network," in *IEEE International Conference on Robotics and Automation*, vol. 3, 2000, pp. 2302–2309.

[150] P. Fröhlich, I. de Almeida Móra, W. Nejdl, and M. Schroeder, "Diagnostic agents for distributed systems," in *ModelAge Workshop*, J.-J. C. Meyer and P.-Y. Schobbens, Eds., vol. 1760. Springer, 1997, pp. 173–186.

[151] N. Roos, A. ten Teije, and C. Witteveen, "A protocol for multi-agent diagnosis with spatially distributed knowledge," in *Proceedings of the AAMAS*. ACM, 2003, pp. 655–661.

[152] M. Kalech, G. A. Kaminka, A. Meisels, and Y. Elmaliach, "Diagnosis of multi-robot coordination failures using distributed CSP algorithms," in *American Association for Artificial Intelligence*. AAAI Press, 2006.

[153] M. Kalech and G. A. Kaminka, "Towards model-based diagnosis of coordination failures," in *American Association for Artificial Intelligence*, M. M. Veloso and S. Kambhampati, Eds. AAAI Press; AAAI Press / The MIT Press, 2005, pp. 102–107.

[154] A. G. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. Boca Raton: CRC Press, 1998.

[155] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete event systems," *Discrete Event Dynamic Systems*, vol. 10, no. 1-2, pp. 33–86, 2000.

[156] M. Ceraolo, "New dynamical models of lead-acid batteries," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1184–1190, Nov. 2000.

[157] S. Barsali and M. Ceraolo, "Dynamical models of lead-acid batteries: Implementation issues," *IEEE Transactions on Energy Conversion*, vol. 17, no. 1, pp. 16–23, Mar. 2002.

[158] E.-J. Manders, G. Biswas, J. Ramirez, N. Mahadevan, J. Wu, and S. Abdelwahed, "A model-integrated computing tool-suite for fault adaptive control," in *Proceedings of the Fifteenth International Workshop on Principles of Diagnosis*, June 2004.

[159] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, "Model-integrated development of embedded software," in *Proceeding of the IEEE*, vol. 91, no. 1, Jan. 2003, pp. 145–164.

[160] MATLAB/Simulink. [Online]. Available: http://www.mathworks.com/