FAULT DIAGNOSIS IN
DISCRETE–EVENT AND HYBRID SYSTEMS

by

Shahin Hashtrudi Zad

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright by Shahin Hashtrudi Zad 1999
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-45687-0
Fault Diagnosis in Discrete–Event and Hybrid Systems
Doctor of Philosophy
1999
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

A framework for on-line passive fault diagnosis in discrete-event systems is proposed. In this approach, the system and the diagnoser (the fault detection system) do not have to be initialized at the same time, and no information about the state or even the condition (failure status) of the system before the initiation of diagnosis is required.

First, a state-based approach for fault diagnosis in finite-state automata is presented. The design of the fault detection system has, in the worst case, exponential time complexity. A model reduction scheme with polynomial time complexity is introduced to reduce the computational complexity of the design. Next the use of timing information to improve the accuracy of diagnosis is considered. Instead of directly extending the framework to timed discrete-event systems, an alternative approach is taken which leads to significant reduction in on-line computing requirements, and in many cases, in the size of the diagnoser at the expense of more off-line design calculations. The issue of diagnosability of failures in this framework is also studied and necessary and sufficient conditions for diagnosability are derived.

In addition, the cases where the discrete–event models used in fault diagnosis are abstractions of hybrid automata are studied. Here, the important issue is whether the discrete–event model contains enough information about the system for the purpose of fault diagnosis. In this regard, two different notions of consistency between high–level (discrete–event) and low–level (hybrid) models are introduced. Sufficient conditions for consistency are derived and a semi-algorithmic method for constructing suitable high–level discrete–event models from low–level hybrid systems is developed.
Acknowledgement

I would like to thank my supervisor, Professor Raymond Kwong, for his invaluable support and guidance throughout this work. Many thanks to Professor W. Murray Wonham whose insightful comments and observations have been extremely helpful.

Special thanks to my friends in the Systems Control Group, in particular Peyman Gohari and Steve Postma, for interesting and inspiring discussions.

This work was supported in part by AlliedSignal Aerospace Canada. Their support is gratefully acknowledged.

I would like to express my deepest gratitude to my family, especially my parents, for their love and support. This work is dedicated to them.
## Contents

1 Introduction ........................................ 1  
   1.1 Thesis Outline ........................................ 5  

2 Fault Diagnosis in Finite-State Automata ................. 9  
   2.1 Plant Model ........................................ 9  
   2.2 Diagnoser ........................................ 13  
   2.3 Diagnosability ...................................... 19  
   2.4 Model Reduction .................................... 30  
   2.5 Simultaneous Failures ............................... 37  

3 Fault Diagnosis in Timed Discrete–Event Systems ......... 44  
   3.1 Plant Model ........................................ 45  
   3.2 Diagnoser ........................................ 58  
   3.3 Time–Diagnosability ............................... 88  

4 Fault Diagnosis and Consistency in Hybrid Systems ... . 99  
   4.1 Plant Model ........................................ 100  
   4.2 Diagnoser ........................................ 109  
   4.3 Diagnosability and Consistency .................... 112  
   4.4 Consistent Model Reduction ....................... 114  

5 Conclusion ........................................... 134  
   5.1 Summary ........................................... 134  
   5.2 Future Research .................................... 136
List of Figures

2.1 Example 2.1. A heating system. ........................................ 11
2.2 Example 2.1. DES model of the heating system. ................. 12
2.3 System and diagnoser. .................................................. 13
2.4 $z_{k+1} = \zeta(z_k, y_{k+1})$ .......................................... 14
2.5 Example 2.1. Diagnoser. ............................................... 17
2.6 Example 2.1. Part of the diagnoser. ................................ 17
2.7 Single-failure scenario with permanent failure modes. ........... 20
2.8 Example of an $F$-uncertain cycle which is not $F$-indeterminate 23
2.9 With $z_0 = \{0\}$, $F$ will be diagnosable. ....................... 25
2.10 Example 2.2. Pump and Valve. ....................................... 27
2.11 Example 2.2. DES models of components. ........................ 27
2.12 Example 2.2. DES model of the pump and valve system. .......... 28
2.13 Example 2.2. Diagnoser. ............................................. 29
2.14 $P z_k = \bar{z}_k$ .......................................................... 32
2.15 Simultaneous occurrence of permanent failures. ................. 38
2.16 Example 2.3. DES model of the pump and valve system. ........ 41

3.1 TDES and the corresponding timed FSMA. .......................... 53
3.2 Example 3.1. A simple TDES. ........................................ 55
3.3 Example 3.1. Timed FSMA. ............................................ 55
3.4 Example 3.2. A neutralization process. ............................ 56
3.5 Example 3.2. Activity transition graph. ............................ 59
3.6 Example 3.2. TDES model. ............................................ 60
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7</td>
<td>Example 3.2. Timed FSMA (Part 1)</td>
<td>62</td>
</tr>
<tr>
<td>3.8</td>
<td>Example 3.2. Timed FSMA (Part 2)</td>
<td>63</td>
</tr>
<tr>
<td>3.9</td>
<td>System and diagnoser</td>
<td>65</td>
</tr>
<tr>
<td>3.10</td>
<td>Example 3.1. Diagnoser</td>
<td>69</td>
</tr>
<tr>
<td>3.11</td>
<td>Example 3.2. Diagnoser</td>
<td>72</td>
</tr>
<tr>
<td>3.12</td>
<td>Using predictions in fault diagnosis</td>
<td>75</td>
</tr>
<tr>
<td>3.13</td>
<td>Example 3.1. $G'_r$.</td>
<td>79</td>
</tr>
<tr>
<td>3.14</td>
<td>Proof of Thm. 3.6. Case 1</td>
<td>84</td>
</tr>
<tr>
<td>3.15</td>
<td>Proof of Thm. 3.6. Case 2. $\tau'$ stands for complex $\tau$ transition and $\sigma \neq \tau'$.</td>
<td>85</td>
</tr>
<tr>
<td>4.1</td>
<td>Evolution of the state of the hybrid system.</td>
<td>101</td>
</tr>
<tr>
<td>4.2</td>
<td>Examples of boundaries: $B_{i,2} = {(\xi_1, \xi_2)</td>
<td>\xi_1 = c &amp; b &lt; \xi_2 &lt; a}$ and $B_{1,2,3,4} = {(b, c)}$.</td>
</tr>
<tr>
<td>4.3</td>
<td>Hybrid system before transformation</td>
<td>105</td>
</tr>
<tr>
<td>4.4</td>
<td>Case 1. Hybrid system after transformation.</td>
<td>105</td>
</tr>
<tr>
<td>4.5</td>
<td>Case 2. Hybrid system after transformation.</td>
<td>106</td>
</tr>
<tr>
<td>4.6</td>
<td>Example 4.1. Water tank.</td>
<td>107</td>
</tr>
<tr>
<td>4.7</td>
<td>Example 4.1. Controller.</td>
<td>107</td>
</tr>
<tr>
<td>4.8</td>
<td>Example 4.1. Hybrid automaton.</td>
<td>108</td>
</tr>
<tr>
<td>4.9</td>
<td>Example 4.1. Transformed hybrid automaton.</td>
<td>109</td>
</tr>
<tr>
<td>4.10</td>
<td>Example 4.1. Diagnoser.</td>
<td>111</td>
</tr>
<tr>
<td>4.11</td>
<td>Hierarchy of models and diagnosers.</td>
<td>114</td>
</tr>
<tr>
<td>4.12</td>
<td>$\Pi^* = \emptyset$.</td>
<td>116</td>
</tr>
<tr>
<td>4.13</td>
<td>$P_{z_k} = \tilde{z}_k$.</td>
<td>119</td>
</tr>
<tr>
<td>4.14</td>
<td>Example 4.1. High-level model.</td>
<td>124</td>
</tr>
<tr>
<td>4.15</td>
<td>Example 4.1. High-level diagnoser.</td>
<td>125</td>
</tr>
<tr>
<td>4.16</td>
<td>Example 4.2. Hybrid automaton.</td>
<td>129</td>
</tr>
<tr>
<td>4.17</td>
<td>Example 4.2. State trajectories of the hybrid system.</td>
<td>129</td>
</tr>
<tr>
<td>4.18</td>
<td>Example 4.2. The high-level DES $\hat{G}_1$.</td>
<td>130</td>
</tr>
<tr>
<td>4.19</td>
<td>Example 4.2. The high-level DES $\hat{G}_2$.</td>
<td>131</td>
</tr>
</tbody>
</table>
4.20 Neighbouring blocks $B_1$ and $B_2$. 

vii
List of Tables

2.1 Example 2.1. Reachability transition system. ...................... 16
2.2 Example 2.2. Reachability transition system. .................. 28
2.3 Example 2.1 Reduced reachability transition system. ......... 34
2.4 Example 2.3. Reachability transition system. .................. 42
2.5 Example 2.3. Diagnoser. ........................................ 42

3.1 Example 3.2. Events and their time bounds. ..................... 57
3.2 Example 3.2. Output and condition maps of TDES. ........... 61
3.3 Example 3.1. Timed reachability transition system. .......... 68
3.4 Example 3.2. Timed reachability transition system. .......... 71

4.1 Example 4.1. Output map. ...................................... 108
4.2 Example 4.1. Reachability transition system. .................. 112
4.3 Example 4.1. Reachability transition system for the DES model. ... 125
Chapter 1

Introduction

Fault detection systems play a crucial role in protecting life and property, and in increasing operational time and productivity. Therefore, they are of paramount importance in aerospace, manufacturing and process industries. A significant portion of the "control" code in real-world systems (up to 80% in some cases) is dedicated to test and diagnosis [9]. Solving diagnostic problems for complex systems is a complicated task requiring a systematic approach. Systematic diagnosis code generation is less prone to human error than manual code generation. In addition to enhancing the reliability and accuracy of diagnosis, the use of systematic approaches can potentially result in faster design process and in reduction in the cost of future revisions and maintenance of diagnosis code. As a result, fault diagnosis has been the subject of extensive research (see, e.g., [12], [21], [26], [27]).

In this work, a failure (fault) \(^1\) refers to a nonpermitted deviation in the behaviour of the system under supervision (or a component of the system) for a bounded or unbounded period of time under specified operating conditions. A valve becoming stuck-closed, decrease in the efficiency of a heat exchanger, bias in the output of a sensor, and leakage in pipelines are examples of failures. If after the occurrence of a failure, the system remains in the faulty condition indefinitely, then the failure is called \textbf{permanent (hard)}. Otherwise, it is \textbf{nonpermanent}. A broken shaft in a

\(^1\)In this thesis, "failure" and "fault" mean the same and are used interchangeably. In the literature, however, they can have slightly different meanings [21].
motor is an example of a permanent failure and a loose wire could be the source of nonpermanent failures in an electrical system.

A typical fault diagnosis system uses the outputs of the sensors of the system under supervision to detect the failure and (if necessary) isolate (locate) the source of failure. Once a failure is detected, a decision has to be made as to how it should be accommodated. Failure accommodation may include a reconfiguration of the (control) system. Here, we address only the problem of fault diagnosis and leave the issue of failure accommodation for future research.

In this thesis, we examine fault diagnosis in systems that, for diagnostic purposes, can be modelled as discrete-event systems (DES). We consider two types of failure: (i) drastic failures, such as “valve stuck-closed” and “sensor short-circuited”, and (ii) partial failures that result in change in timing characteristics, such as “valve sticking”. Other types of partial failures such as drift in sensor or small changes in dynamics of actuators are not considered here because they are difficult to model in a DES framework.

We concentrate on on-line passive diagnosis only; i.e., the system under supervision is operational and the fault detection system does not use test inputs for diagnosing failures but relies solely on output signals.

One of the commonly-used techniques for fault diagnosis is “hardware redundancy” in which multiple sensors (usually three) are used for measuring each plant variable. The outputs of the sensors measuring the same variable are compared to determine the plant variable and to detect sensor failures. This method is simple and fairly reliable, though expensive and of no use in detecting common-cause failures. Moreover, it is only suitable for detecting sensor failures.

Expert systems are also used for diagnosing failures. These systems use the experience and knowledge of experts (stored as a set of rules) and an inference engine to diagnose failures. Expert systems are particularly useful in cases where it is hard to obtain a model for the plant. Gathering the expertise and knowledge required for building an expert system is usually not an easy task. Furthermore, there will be no guarantee for the completeness of the resulting diagnostic rules.
In the above techniques, the plant’s model is not used in diagnosis. In addition to model-free methods, several model-based techniques for fault diagnosis have been proposed in the literature. In a model-based method, the observed behaviour of the plant is compared with that expected from the plant’s model. Based on this comparison, an inference is made about the condition (normal/faulty) of the plant. It should be noted that obtaining a model for the normal and faulty behaviours of a system could be difficult (if not impossible) and time consuming. Here it is assumed that all of the different ways in which the system can fail are known. Finding these failure modes is perhaps the most challenging problem in hazard analysis. Systematic model-based diagnostic techniques do not directly help us with this problem. However, at least they guarantee that none of the known failure modes will be left out in the design process.

Some of the model-based techniques which are suitable for diagnosing failures in discrete-event systems are described in the following. By far, fault-tree analysis [19] seems to be the most popular approach. Fault trees can be synthesized automatically. They provide a pictorial display of the system which can be easily read and understood. However, they have certain limitations: it is difficult to incorporate information about ordering and timing of events in a fault tree [21]; moreover, problems arise in the analysis of systems that go through more than one phase of operation [20]; also, there seems to be no way of treating common-cause failures resulting from fault propagation (domino effects) using fault trees [20].

In [14], for failure detection, the timed event sequence generated by the DES under supervision is compared with a set of specifications for normal operation, called templates. This technique, which relies on sequencing and timing relationships, is suitable for fault detection in certain high-speed manufacturing lines where the plant can be modelled as a set of an unspecified number of timed automata operating in parallel and independent of each other. It is shown [14], [25] that under certain restrictions, templates can be used for fault detection. Failure isolation and diagnosability have not been addressed in [14] and [25].

In [28] and [17], the dynamics of the plant under supervision is modelled with
state-space equations, with components of the state vector belonging to finite sets. The state equations are described using predicates on the components of the state vector and inputs to the system.

In [17], fault detection and isolation has been examined under the strong assumption that changes in the state of the system are observable. A fault is called detectable if there exist values for state and input that would lead to the detection of the fault in a finite number of steps. This definition of detectable faults, and the corresponding definition for isolatable faults, seem to be suitable for active diagnosis. Algorithms for the verification of detectability and computation of the inputs required for fault detection are obtained under the strong assumption that failures only affect sensor maps (and not the dynamics of the system's state). Failure isolation has also been studied under the same assumptions. Moreover, a procedure for partitioning faults into sets of indistinguishable faults has been derived. Compared with the descriptions based on state transition graphs, representations of the system's dynamics and the diagnosis algorithms with predicates could be more compact, though somewhat difficult to interpret. However, they could potentially result in efficient computer code for diagnosis system design.

Finite-state automata have also been used in the study of diagnostic problems. The use of finite-state automata and the corresponding state transition diagrams make the design and maintenance of complex control and diagnostic systems easier (see, e.g., [11]) and it appears that these models match the internal models many people use in trying to analyze and understand complex systems (Ch. 14 of [21]). It is also straightforward to capture the ordering of events using finite-state automata. A problem associated with using automata is the computational complexity.

In [22], a state-based approach is used for the study of fault diagnosis. In a state-based approach, it is assumed that the state set of the system can be partitioned according to the condition (failure status) of the system. The problem of fault diagnosis is to use measurements (sensor readings) to determine the partition that the state belonged to (hence, the condition) at the time the last measurement was received. In [22], first an off-line diagnosis problem is studied where the state of the
system is assumed fixed during diagnostic tests, and output measurements are used for failure detection and isolation. The concept of testability is also introduced and studied. An active on-line diagnosis problem has been studied in [22]. In this case, input test sequences for fault diagnosis are computed. Furthermore, the notion of on-line diagnosability is introduced.

In [29] and [30], passive on-line diagnosis is studied using an event-based framework: based on observed events, inference is made about the occurrence of (unobservable) failure events. Diagnosability of failures is also studied. In this approach, a failure is called diagnosable if it can be detected and isolated after the occurrence of at most a bounded number of events following the failure. This framework has been extended to utilize information about the timing of events [4]. Integrated approaches to fault diagnosis and supervisory control have also been studied in an event-based framework in [5] and [31].

We note that testing of finite-state machines [18] is related to fault diagnosis. However, the framework used for that purpose is different: the finite-state machines are usually assumed to be deterministic with a fixed condition (failure status); also it is assumed that transitions can always be observed even if they do not result in a change in output. These assumptions often do not hold in fault diagnosis of control systems.

1.1 Thesis Outline

In Chapter 2, we present a state-based approach for passive on-line diagnosis of finite-state automata. The resulting fault diagnosis system, also called diagnoser, is a finite-state machine that takes the output sequence of the system as input and generates at its output an estimate of the condition (failure status) of the system. The diagnoser, as a finite-state machine, can be automatically translated into computer code. In this way, code generation becomes automated. There are similarities between our work and the event-based approach of [29]. However, our framework seems to us to be simpler and the diagnoser is easier to compute. Moreover, we do not assume
that the plant and the diagnoser are initialized at the same time. Knowledge of the state and even the condition of the plant (normal/faulty) at the time the diagnoser is initialized are not required. This means that our method can be used for diagnosing failures that could have occurred before the initiation of diagnosis. We show that any problem in an event-based framework can be recast as a problem in the state-based framework presented here. We also introduce a model reduction scheme with polynomial time complexity to reduce the computational complexity of designing the diagnoser, which has exponential time complexity in the worst case. To our knowledge, the use of model reduction in the design of a diagnoser has not been studied previously in the DES literature.

We also study failure diagnosability. In our framework, a failure is diagnosable if it can be detected and isolated after the occurrence of at most a bounded number of events following both the occurrence of failure and initialization of the diagnoser. Note that in our framework, the diagnosis might be initiated after the occurrence of failure. We obtain necessary and sufficient conditions for diagnosability.

The diagnoser of Chapter 2 uses changes in the output sequence to detect and isolate failures. In many cases, however, a failure changes only the timing of the output sequence (rather than the sequence). In other cases, as a result of failure, the system stops generating new output symbols. In these situations, timing information can be used to increase the accuracy and speed of diagnosis. In Chapter 3, we extend our state-based framework to incorporate timing information. We assume that the system can be modelled as a timed discrete-event system (TDES). Timed discrete-event systems can be used for capturing timing information in a useful range of problems in control engineering in a reasonably simple fashion [3]. This is accomplished by describing the sequence of events occurring in the system with respect to the ticks of a global clock. For the purpose of fault diagnosis in TDES, it is possible to adopt our diagnosis theory for finite-state automata simply by treating the clock tick as an extra output signal. In [4], a similar method has been developed for the extension of the event-based framework of [29]. This straightforward approach could result in very large state sets for the corresponding diagnosers. In this work, however,
we have taken an alternative approach in which the process of updating the estimate of the system's condition is performed only when a new output symbol is generated. The update process uses the generated output symbols and the number of clock ticks between them. No update at clock ticks is required in this method. For fault diagnosis between two consecutive output symbols, we rely on predictions of the system's condition. This results in significant reduction in on-line computing requirements, and in many cases, in the size of the diagnoser, at the expense of extra off-line design calculations. Moreover, we study failure diagnosability in TDES, introduce the concept of time-diagnosability in our framework and derive necessary and sufficient conditions for time-diagnosability.

In Chapter 2, it is assumed that the plant under supervision can be modelled, for the purpose of fault diagnosis, as a discrete-event system. In many practical cases, the DES may be a simplified model (i.e., abstraction) of a hybrid automaton (See [15], [2] and references therein). A hybrid automaton is a system whose dynamics is described by a combination of differential (or difference) equations and a state transition graph. In other words, it is a blend of continuous-variable and discrete-event models. In fact, the state vector of a hybrid automaton contains both continuous and discrete components. The continuous-variable part of the model describes the evolution of the continuous components of the state vector while the discrete-event part models changes in the dynamics of the continuous part (resulting from failure for example). Examples of these systems can be found in flexible manufacturing, process control and aerospace systems.

Typically, the design of a diagnosis system, which is based on a simplified discrete-event model, is verified through simulation and test under a large (though finite) number of operating conditions of the hybrid system. In Chapter 4, we consider the development of a more systematic approach to the above verification problem. Towards this end, we first extend our framework for fault diagnosis in DES to hybrid systems. This framework forms a basis for examining the question of whether a high-level DES contains enough information about the low-level hybrid model, which in turn, will lead to the introduction of the notion of consistency between the high-
level and low-level models. The high-level and low-level models are called consistent if the analysis and design based on the high-level model yields the same results as that performed at the low level. We then provide a set of sufficient conditions for consistency. The above notion of consistency is somewhat restrictive and, in fact, too strong. In order to be able to build suitable high-level (DES) models for a larger class of hybrid automata, we introduce the notion of weak consistency. This will result in a semi-algorithmic method for obtaining suitable high-level models. This technique consists of a trial-and-error sequence involving building high-level models. At each step, a weakly consistent high-level model is built and checked if it is suitable for fault diagnosis. Weakly consistent models are easier to construct than consistent ones. Unfortunately, in general, no guarantee exists for the finite termination of the above design sequence.

Finally, in Chapter 5, we present a summary of our results and discuss directions for future research.
Chapter 2

Fault Diagnosis in Finite-State Automata

In this chapter, we propose a framework for diagnosing failures in finite-state automata. Many systems in aerospace, manufacturing and process industries can be modelled, for diagnostic purposes, as finite-state automata. In Chapters 3 and 4, we will extend this approach to timed discrete-event and hybrid systems.

We discuss failure modelling in Section 2.1. In Section 2.2, we introduce the diagnoser and in Section 2.3, define the notion of diagnosability of failures in our framework and derive necessary and sufficient conditions for failure diagnosability. Section 2.4 presents a model reduction scheme for reducing the computational complexity of the diagnoser design. The discussion in Sections 2.1 to 2.4 assumes that at most one failure mode may occur at a time. In Section 2.5, the results of Sections 2.1 to 2.4 will be extended to the case of simultaneous occurrence of two (or more) failure modes.

2.1 Plant Model

We assume that the plant under control, i.e., plant along with low-level continuous controllers and DES supervisors, can be modelled as a nondeterministic finite-state Moore automaton (FSMA) $G = (X, \Sigma, \delta, x_0, Y, \lambda)$, where $X$, $\Sigma$ and $Y$ are the finite
state, event and output sets; \(x_0\) is the initial state, \(\delta : X \times \Sigma \to 2^X\) the transition function and \(\lambda : X \to Y\) the output map (\(2^X\) denotes the power set of \(X\)).

The model describes the behaviour of the system in both normal (system functioning properly) and faulty situations. Suppose there are \(p\) failure modes \(F_1, \ldots, F_p\). Each failure mode corresponds to some kind of failure in an instrument (valve, sensor, etc.) or a set of such failures. The event set \(\Sigma\) includes failure events. In Sections 2.1 to 2.4, we assume at most one of the failure modes may occur at a time. This means that the system can be in only one of the following \(p + 1\) conditions: \(N\) (normal), \(F_1, \ldots, F_p\). We call this single-failure scenario and it should not be confused with single failure mode situation in which the system has only one failure mode, i.e., \(p = 1\). Simultaneous occurrence of two (or more) failure modes will be discussed in Section 2.5.

Let \(\mathcal{K} := \{N, F_1, \ldots, F_p\}\) denote the condition set of the system. It is assumed that the state set \(X\) can be partitioned according to the condition of the system: \(X = X_N \cup X_{F_1} \cup \cdots \cup X_{F_p}\) (\(\cup\) denotes disjoint union). Define the condition map \(\kappa : X \to \mathcal{K}\) such that for every \(x \in X\), \(\kappa(x)\) is the condition of the system at the state \(x\): \(\kappa(x) = N\) if \(x \in X_N\), and \(\kappa(x) = F_i\) if \(x \in X_{F_i}\) (\(i \in \{1, \ldots, p\}\)). Also (abusing notation) extend the definition of \(\kappa\) to the subsets of \(X\): \(\kappa(z) = \cup \{\kappa(x) \mid x \in z\}\), for any \(z \subseteq X\).

In failure detection and isolation, given the output sequence \((y_1y_2y_3\ldots)\), we want to find the condition of the system. Note that only changes in the output are assumed to be observable. This means that a transition of the system from one state to another state having the same output will not be noticed, i.e., the transition will be unobservable. So in the output sequence: \(y_i \neq y_{i+1}\) for \(i \geq 1\).

In general, some of the events in \(\Sigma\) are observable. If the occurrence of these observable events cannot be inferred from the output sequence, then the information about the occurrence of the observable events can be included in the output map in the following way. Consider a transition \(x_1 \xrightarrow{\sigma} x_2\) with \(\sigma\) being observable. Let us assume that the occurrence of \(\sigma\) cannot be inferred from the output; for example, suppose \(\lambda(x_1) = \lambda(x_2)\). Then replace the transition with the two transitions \(x_1 \xrightarrow{\sigma} x'_1 \xrightarrow{\lambda} x_2\).
where \( x'_1 \) and \( \epsilon \) are new state and unobservable event that have to be added to the state and event set of the system. The output set \( Y \) must be extended to \( Y \times \{0, 1\} \) and the output at any \( x \neq x'_1 \) should be redefined as \( (\lambda(x), 0) \) and at \( x'_1 \) as \( (\lambda(x_1), 1) \). Therefore the \( \sigma \)-transition from \( x_1 \) to \( x_2 \) shows up at the output as change of the output from \( (\lambda(x_1), 0) \) to \( (\lambda(x_1), 1) \) and from \( (\lambda(x_1), 1) \) to \( (\lambda(x_2), 0) \). The above procedure can be repeated for other observable events (if necessary). From now on, it will be assumed that information about the occurrence of the observable events has already been transferred and included in the output map.

**Example 2.1 - Heating System**

A heating system uses a heater, a temperature sensor and an ON/OFF controller to regulate the temperature of a room about a set-point (Fig. 2.1). The DES model is shown in Fig. 2.2. The temperature can be low ‘l’, below but close to set-point ‘b’, or above set-point ‘a’. For example, if the set-point is 22°C, then ‘a’ corresponds to the temperatures above 22°C, ‘b’ to the temperatures between, say, 20°C and 22°C, and ‘l’ to the temperatures below 20°C. The controller can turn the heater on (enable it) or turn it off (disable it). It is assumed that the heater may fail and remain off (even if it is turned on). Each dashed arc in Fig. 2.2 represents a heater-failure event. ‘Load’ models the effect of disturbance such as the temperature of the adjoining room and the ambient temperature, and is supposed to have two states ‘normal (n)’ and ‘above normal (a)’. It is also assumed that even when the load is above normal, the heater can keep the temperature close to the set-point. In the heating system: \( X = \{1, 2, \ldots, 24\} \), \( Y = \{ld, le, bd, be, ad, ae\} \), \( \mathcal{K} = \{N, F\} \), \( \kappa(i) = N \) for \( 1 \leq i \leq 12 \), and \( \kappa(i) = F \) for \( 13 \leq i \leq 24 \). The output symbols are
explained below:

- ld temperature low, heater turned OFF (disabled)
- le temperature low, heater turned ON (enabled)
- bd temperature below set-point, heater turned OFF (disabled)
- be temperature below set-point, heater turned ON (enabled)
- ad temperature above set-point, heater turned OFF (disabled)
- ae temperature above set-point, heater turned ON (enabled).

Note that the output symbol may contain information about the commands issued by the DES supervisor (in this case, the ON/OFF controller), in addition to sensor readings.

The system has 24 states. For example, in state 1, load is above normal, temperature is low, and the heater is turned off and in the normal mode. At this point, the controller turns the heater on and the system moves from the state 1 to 3. If the load becomes normal, then the system moves to the state 4. The rest of Fig. 2.2 can be interpreted similarly.
2.2 Diagnoser

A diagnoser is a system that detects and isolates failures. In our framework, it is a finite-state machine that takes the output sequence of the system \((y_1 y_2 \cdots y_k)\) as input and generates at its output an estimate of the condition of the system at the time that \(y_k\) was generated. Specifically, based on the output sequence up to \(y_k\), a set \(z_k \in 2^X - \{0\}\) is calculated to which \(x\) must belong at the time that \(y_k\) was generated. \(\kappa(z_k)\) will be the estimate of the system's condition (Fig. 2.3). Upon observing \(y_{k+1}\), \(z_k\) will be updated.

Before formally defining the diagnoser, we introduce the concept of output-adjacency which we will find useful in diagnoser design.

**Definition 2.1** For any two states \(x, x' \in X\), we say \(x'\) is **output-adjacent** to \(x\) and write \(x \Rightarrow x'\) if \(\lambda(x) \neq \lambda(x')\) and there exist \(l \geq 2\), the states \(x_2, \ldots, x_{l-1}\) (if \(l > 2\)), and the events \(\sigma_1, \ldots, \sigma_{l-1}\) such that \(x_{i+1} \in \delta(x_i, \sigma_i)\) and \(\lambda(x_i) = \lambda(x)\), for all \(1 \leq i \leq l - 1\), with \(x_1 = x\) and \(x_l = x'\).

This means that \(x'\) is output-adjacent to \(x\) if \(x\) and \(x'\) have different outputs and \(x'\) can be reached from \(x\) using a path \(^1\) along which the output is \(\lambda(x)\).

We define the **diagnoser** to be a finite-state Moore machine \(D = (Z \cup \{z_0\}, Y, \zeta, z_0, \mathcal{K}, \kappa)\), where \(Z \cup \{z_0\}\), \(Y\) and \(\mathcal{K} \subseteq 2^X - \{0\}\) are the state, event and output sets of \(D\); \(z_0 := (z_0, 0)\) is the initial state, with \(z_0 \in 2^X - \{0\}\); \(Z \subseteq 2^X - \{0\}\),

\(^1\)Suppose for a sequence of states \(x_1, \ldots, x_n\) and events \(\sigma_1, \ldots, \sigma_n \in \Sigma\), with \(n \geq 2\), we have \(x_{i+1} \in \delta(x_i, \sigma_i)\), for \(1 \leq i \leq n - 1\). Then \(x_1 \xrightarrow{\sigma_1} x_2 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_{n-1}} x_n\) is called a **path** in (the state transition graph of) the system.
and \( \zeta : Z \cup \{z_0\} \times Y \rightarrow Z \) is the transition (partial) function; \( \kappa : Z \cup \{z_0\} \rightarrow \hat{K} \) is the output map, with the definition of \( \kappa \) extended according to: \( \kappa(z_0) := \kappa(z_0) \). Each diagnoser state \( z \), except \( z_0 \), is identified with a nonempty subset of \( X \). However, \( z_0 \) is considered to be different from other states of the diagnoser because the update law \( \zeta \) at \( z_0 \) is different from that at other states. To distinguish \( z_0 \) from other states, it has been identified with a pair \((z_0, 0)\) (rather than a subset).

The diagnoser state transition \( z_{k+1} = \zeta(z_k, y_{k+1}) \) is given by:

\[
\begin{align*}
   z_1 &= z_0 \cap \lambda^{-1}(\{y_1\}) \\
   z_{k+1} &= \{x \mid \lambda(x) = y_{k+1} \land (\exists x' \in z_k : x' \Rightarrow x)\} \quad \forall k \geq 1.
\end{align*}
\]

Therefore \( z_{k+1} \) will be the set of states, having output \( y_{k+1} \), that are reachable from the states in \( z_k \) using paths along which the output is \( y_k \) (Fig. 2.4).

\( z_0 \) contains the information available about the state of the system at the time that the diagnoser is initialized, before the reading of sensors begins. Usually, \( z_0 = X \), because the diagnoser may be initialized at any time while the system is in operation and in this situation the state of the system is not known exactly. If the system and the diagnoser are initialized at the same time, then \( z_0 = \{x_0\} \). If the system is only known to be normal at the time that the diagnoser is initialized, then \( z_0 = X_N \).

Suppose \( z^1, z^2 \in Z \) and \( \zeta(z^1, y) = z^2 \) for some \( y \in Y \). Given \( z^1 \), in order to compute \( z^2 \) we have to find for every \( x_1 \in z^1 \), all \( x_2 \) such that \( \lambda(x_2) = y \) and \( x_1 \Rightarrow x_2 \). Since every \( x \in X \) typically belongs to several states of the diagnoser, it is computationally economical to compute the set of output-adjacent states, for every \( x \in X \). This can be done in \( \mathcal{O}(|X|^2 + |X||T|) \) time because a breadth-first search
reachability analysis for each \( x \in X \) can be done in \( \mathcal{O}(|X| + |T|) \) time [6]. Here \(|X|\) and \(|T|\) are the cardinalities of \( X \) and \( T \) (the set of transitions of \( G \)).

We store the information about the output–adjacent states in the **reachability transition system** (RTS). We define the RTS (corresponding to \( G \)) to be the transition system \( \tilde{G} = (X, R, Y, \lambda) \), which has \( X, Y \), and \( \lambda \) as the state set, output set and output map. \( R \subseteq X \times X \) and \((x_1, x_2) \in R \) if and only if \( x_1 \Rightarrow x_2 \). As mentioned before, \( \tilde{G} \) can be computed in \( \mathcal{O}(|X|^2 + |X||T|) \) time. With \( \tilde{G} \) available, one can compute the diagnoser. Later we will see that \( \tilde{G} \) is also useful in on-line implementation of diagnostic computations and in model reduction.

**Example 2.1 - Heating System** (Cont’d)

The RTS (in the form of a table) and the diagnoser for the heating system are given in Table 2.1 and Fig. 2.5 (assuming \( z_0 = X \)). To see how the diagnoser works, suppose the first output symbol \( y_1 \) is ‘be’. Then \( z_1 = \{5, 6, 17, 18\} \). If the next output symbol \( y_2 \) is ‘le’, then \( z_2 = \{15, 16\} \). Using the RTS (Table 2.1), the reader can verify that each of the states in \( z_2 \), namely, 15 and 16, is output–adjacent to (at least) one state in \( z_1 \). If instead of ‘le’, \( y_2 = ‘ae’ \), then \( z_2 = \{7, 8\} \). The rest of Fig. 2.5 can be similarly interpreted.

In this example, heater failure can be detected using output observations unless it occurs while the temperature is low. For example, suppose that the diagnoser is initialized when the system is at the state 10. The output at this state is ‘ad’. Therefore \( z_1 = \{9, 10, 21, 22\} \) and \( \kappa(z_1) = \{N, F\} \). Now assume that the state \( x \) moves to 12, and then to 6; at this point a heater failure occurs following which the state moves to 18 and finally to 16. While the system evolves along this path, the output sequence ‘bd, be, le’ will be generated which will take the diagnoser to the state \( z_4 = \{15, 16\} \). Since \( \kappa(z_4) = \{F\} \), the diagnoser eventually detects the failure.

To see why heater failure cannot be detected while the temperature is low, suppose the system is at the state 3 when the diagnoser is started. At this state the output is ‘le’; thus \( z_1 = \{3, 4, 15, 16\} \) and \( \kappa(z_1) = \{N, F\} \). A heater failure at this point takes the state \( x \) to 15. No new output symbol will be generated following the failure or
Table 2.1: Example 2.1. Reachability transition system.

<table>
<thead>
<tr>
<th>State</th>
<th>Output-adjacent states (output)</th>
<th>State</th>
<th>Output-adjacent states (output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3,4,15,16 (le)</td>
<td>13</td>
<td>15,16 (le)</td>
</tr>
<tr>
<td>2</td>
<td>3,4,15,16 (le)</td>
<td>14</td>
<td>15,16 (le)</td>
</tr>
<tr>
<td>3</td>
<td>5,6 (be)</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>5,6 (be)</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>8 (ae) / 15,16 (le)</td>
<td>17</td>
<td>15,16 (le)</td>
</tr>
<tr>
<td>6</td>
<td>8 (ae) / 15,16 (le)</td>
<td>18</td>
<td>15,16 (le)</td>
</tr>
<tr>
<td>7</td>
<td>9,10,21,22 (ad)</td>
<td>19</td>
<td>21,22 (ad)</td>
</tr>
<tr>
<td>8</td>
<td>9,10,21,22 (ad)</td>
<td>20</td>
<td>21,22 (ad)</td>
</tr>
<tr>
<td>9</td>
<td>11,12,23,24 (bd)</td>
<td>21</td>
<td>23,24 (bd)</td>
</tr>
<tr>
<td>10</td>
<td>11,12,23,24 (bd)</td>
<td>22</td>
<td>23,24 (bd)</td>
</tr>
<tr>
<td>11</td>
<td>5,6,17,18 (be)</td>
<td>23</td>
<td>17,18 (be)</td>
</tr>
<tr>
<td>12</td>
<td>5,6,17,18 (be)</td>
<td>24</td>
<td>17,18 (be)</td>
</tr>
</tbody>
</table>

Afterwards. Therefore, \( \{N, F\} \) will remain as the estimate of the system's condition. As a result, the failure will not be detected with certainty. The issue of diagnosability of failures will be discussed in Section 2.3.

The diagnoser, as a finite-state machine, can be automatically translated into computer code. For example, consider Fig. 2.6 in which part of the diagnoser of Fig. 2.5 is shown. The three diagnoser states are labeled \( D_1, D_2 \) and \( D_3 \). Each state of the diagnoser can be translated into a segment of computer code. By putting these segments together, the computer code corresponding to the diagnoser can be automatically generated. A pseudocode for the state \( D_1 \) is given below.

\[
D1 \quad z=[5,6,17,18]; \\
\quad k=['N';'F']; \\
\quad while y=='be' \\
\quad \quad read y \\
\quad end \\
\quad if y=='le' \quad go to D2 \\
\quad \quad elseif y=='ae' \quad go to D3 \\
\quad \quad else \quad go to INCONSISTENCY \\
\quad end
\]
Figure 2.5: Example 2.1. Diagnoser.

Figure 2.6: Example 2.1. Part of the diagnoser.
Note that if the output symbol that follows 'be' is neither 'le' nor 'ae', then an inconsistency between the model and the observed behaviour of the system is detected. This could be the result of either an unforeseen failure mode or a glitch in the sensor output.

Since the number of states of the diagnoser is in the worst case exponential in $|X|$, the whole diagnoser may take up a large amount of computer memory. In this case, it might be better to store the reachability transition system in memory and use it to perform the diagnostic computations on-line \(^2\): having $z_k$ and the observation $y_{k+1}$, use $\hat{G}$ to compute $z_{k+1}$ and update the estimate of the system's condition. $\hat{G}$ contains $|X|$ states and at most $|X|(|X|-1)$ transitions.

There are similarities between our framework and that of [29], especially in the use of an observer for diagnosis. However, while we try to determine the condition of the system for fault detection, in [29] the authors attempt to detect the unobservable failure events. This leads to some differences:

- In our approach, the state and even the condition of the system do not have to be known at the time that the diagnoser is started. The diagnoser can be initialized at any time while the system is in operation, not necessarily when the system is started. If say a failure occurs before the diagnoser is initialized, then our diagnoser can eventually detect the faulty condition and isolate the failure (assuming the failure is diagnosable). However, an event-based diagnoser cannot detect the failure because the failure event has already happened when the diagnoser is initialized.

- Our approach is simpler. In particular, in this framework, computation of the diagnoser is less complex. This is because at each step, after observing a new output symbol, we only have to update our estimate of the system's state $z_k$. In the event-based approach of [29], after the occurrence of an observable event, in addition to updating the state estimate, all of the paths that the state of the

---

\(^2\)In this thesis, “off-line calculations” refers to the computations at the design stage and “on-line calculations” refers to the diagnostic computations performed when the plant is in operation.
system might have evolved along since the occurrence of the previous observable event, have to be checked for the occurrence of failure events. In the reachability analysis required for estimate update in our framework, however, (in general) only a subset of these paths are examined. The simplicity of our framework significantly contributes to the development of transparent results for model reduction (Section 2.4) and fault diagnosis in timed discrete-event and hybrid systems (Chapters 3 and 4).

Remark 2.1 We note that a diagnosis problem in an event-based framework can always be transformed to an equivalent problem in the state-based framework presented in this thesis. More specifically, the problem of detecting a set of unobservable events can be cast into the problem of finding the block of some partition that the state belongs to. To see this, suppose we want to detect an unobservable event $F$. We replace the original automaton $G = (X, \Sigma, \delta, x_0, Y, \lambda)$ with $G^* = (X^*, \Sigma, \delta^*, (x_0, 0), Y, \lambda^*)$, where $X^* = X \times \{0, 1\}$, $\delta^*((x, f), \sigma) = \cup\{(x', f)|x' \in \delta(x, \sigma)\}$ if $\sigma \neq F$, $\delta^*((x, f), F) = \cup\{(x', 1)|x' \in \delta(x, F)\}$, and $\lambda^*((x, f)) = \lambda(x)$ for all $x \in X$ and $f \in \{0, 1\}$. The problem of detecting $F$ in $G$ becomes equivalent to finding whether in $G^*$, $x^* \in X \times \{1\}$ or not. This method can be used for detecting any number of unobservable events. However, it should be noted that the size of the state space of the transformed automaton $G^*$ will increase with the number of the unobservable events.

2.3 Diagnosability

In this section, we study the question of whether or not a failure mode $F_i$ can always be detected and isolated. We assume for now that the failure modes are permanent. Later in Remark 2.4, we will discuss nonpermanent failures. We also assume that only one failure mode may occur at a time (i.e., single-failure scenario). Simultaneous occurrence of failures will be discussed in Section 2.5. The assumption of permanent failure modes implies that in the finite-state automaton model of the system given in Section 2.1, there are no transitions from the states in $X_{F_i}$ ($i \in \{1, \ldots, p\}$) to any state in $X_N \cup (\cup_{j \neq i} X_{F_j})$, i.e., $X_N \cup (\cup_{j \neq i} X_{F_j})$ is not reachable from $X_{F_i}$ (Fig. 2.7).
Figure 2.7: Single-failure scenario with permanent failure modes.

For instance, in Example 2.1 (Heating System), heater failure is permanent and none of the states in $X_N$ is reachable from the states in $X_F$.

**Definition 2.2** A state $z$ of the diagnoser is called $F_i$-certain if $\kappa(z)$, the corresponding estimate of the system's condition, indicates that the failure has occurred. \(\Box\)

In a single-failure scenario, $z$ is $F_i$-certain iff $\kappa(z) = \{F_i\}$.

**Definition 2.3** A state $z$ of the diagnoser is called $F_i$-uncertain if $z$ is not $F_i$-certain but $\kappa(z)$ indicates that the failure $F_i$ might have occurred, i.e., occurrence of $F_i$ is consistent with the observed output sequence. \(\Box\)

In a single-failure scenario, $z$ is $F_i$-uncertain iff $F_i \in \kappa(z)$ but $\kappa(z) \neq \{F_i\}$. For instance, in Fig. 2.5 $z = \{15, 16\}$ is $F$-certain while $z = \{5, 6, 17, 18\}$ is $F$-uncertain. Note that if $z$ is not $F_i$-certain, it does not mean that it is $F_i$-uncertain.

Note that if for some $k_0 \geq 0$ and a failure mode $F_i$, $\kappa(z_{k_0}) = \{F_i\}$, then $\kappa(z_k) = \{F_i\}$ for all $k \geq k_0$ (because $X_N \cup (\bigcup_{j \neq i} X_F)$ is not reachable from $X_{F_i}$). Therefore if the diagnoser reaches an $F_i$-certain state, it will remain in the set of $F_i$-certain states, i.e., it will 'lock' on $F_i$.

In general, the number of events that may take the diagnoser to diagnose a failure (assuming it can) depends on

- the path that the system evolves on, and
- the time of initialization of the diagnoser.
For instance, in Example 2.1, if the system evolves on the path $10 \rightarrow 12 \rightarrow 6 \rightarrow 18 \rightarrow 16$ and the diagnoser is initialized when the system is at the state 10, then the failure will be diagnosed after the system enters the state 16, one event after the occurrence of the failure. If, on the other hand, the system's state moves on $10 \rightarrow 22 \rightarrow 24 \rightarrow 23 \rightarrow 17 \rightarrow 15$ and, as in the previous case, the diagnosis is initiated when the system is at the state 10, then the failure will be diagnosed after 4 events when the system enters the state 15. Of course, in this case, if the diagnoser was initialized after the system entered the state 15 (instead of 10), then the failure would not be detected at all.

We say a failure is diagnosable if it can always be detected and isolated after the occurrence of a bounded number of events, as explained in the following.

**Definition 2.4** A permanent failure mode $F_i$ is **diagnosable** if there exists an integer $N_i \geq 0$ such that following both the occurrence of the failure and initialization of the diagnoser, $F_i$ can be detected and isolated (i.e., the diagnoser reaches an $F_i$-certain state) after the occurrence of at most $N_i$ events in the system.

In the above definition, no assumption is made about the system's condition (normal/faulty) at the time that the diagnoser is initialized, i.e., the diagnoser might have been initialized either before or after the occurrence of the failure. According to Def. 2.4, the heater failure in Example 2.1 (Heating System) is not diagnosable.

**Remark 2.2** In Def. 2.4, a failure is defined to be diagnosable if it can be detected and isolated after the occurrence of at most a bounded number of events. It should be noted that even though an infinite number of events usually takes an infinite time to occur, a finite number of events may take an arbitrarily long time to happen. This issue will further be discussed in Chapter 4.

We need a few more definitions before providing necessary and sufficient conditions for diagnosability. Sometimes generation of a particular output symbol is an indication of the occurrence of a failure mode. A sensor failing open-circuited is an example (assuming open circuit can be detected directly by the electrical interface).
Definition 2.5 If the occurrence of a failure mode $F_i$ can be directly concluded from the generation of an output symbol $y \in Y$, then $y$ is called $F_i$-indicative.

In a single-failure scenario, $y \in Y$ is $F_i$-indicative iff $\lambda^{-1}\{\{y\}\} \subseteq X_{F_i}$.

Definition 2.6 A cycle $^3$ of $F_i$-uncertain states of the diagnoser is called an $F_i$-uncertain cycle.

We shall see later in this section that in one type of undiagnosable failure, the faulty system can generate a periodic output sequence that throws the diagnoser into a fault-uncertain cycle. We call these cycles fault-indeterminate.

Definition 2.7 Suppose $z_1, \ldots, z^m$ is a cycle of $F_i$-uncertain states of the diagnoser. The cycle is called $F_i$-indeterminate if there exist $l \geq 1$ and $x_1^j, x_2^j, \ldots, x_l^j \in z^j$, for all $1 \leq j \leq m$ such that $x_k^j \in X_{F_i}$ for all $1 \leq j \leq m$, $1 \leq k \leq l$ and $x_1^1, x_2^1, \ldots, x_1^m, x_2^1, \ldots, x_2^m, \ldots, x_1^1, \ldots, x_l^m$ form a cycle in the RTS. The RTS cycle is called an underlying faulty cycle of the $F_i$-indeterminate cycle.

According to Def. 2.7, if $z_1, \ldots, z^m$ is an $F_i$-indeterminate cycle, then the system has a cycle in the faulty mode $F_i$ such that when it evolves on the cycle, it will generate the output sequence $\lambda(x_1^1), \ldots, \lambda(x^m_i)$ periodically. The cycle in the mode $F_i$ and the corresponding output sequence can keep the diagnoser in the $F_i$-uncertain cycle $z_1, \ldots, z^m$ indefinitely, and in this case, the failure $F_i$ will not be diagnosed.

Not every fault-uncertain cycle is fault-indeterminate. Consider the FSMA of Fig. 2.8. In this system, $X = \{1, 2, \ldots, 9\}$, $X_N = \{1, 7, 8, 9\}$, $X_F = \{2, 3, 4, 5, 6\}$ and $Y = \{\eta, \alpha, \beta, \gamma, \nu\}$. The diagnoser for this system is given in Fig 2.8, assuming $z_0 = X$. The diagnoser states $\{3, 7\}$, $\{4, 8\}$ and $\{5, 9\}$ form an $F$-uncertain cycle. This cycle is not $F$-indeterminate, however, because the system states 3, 4 and 5 do not form a cycle, i.e., after the occurrence of failure, the output sequence $\alpha\beta\gamma$ will be generated only once, after which $\nu$ will be generated which takes the diagnoser to the $F$-certain state $\{6\}$.

---

$^3$A path $x_1 \xrightarrow{z_1} \cdots \xrightarrow{z_n} x_n$, with $n \geq 2$, is a cycle if $x_1 = x_n$. 

22
Figure 2.8: Example of an $F$-uncertain cycle which is not $F$-indeterminate.

Theorem 2.1 provides necessary and sufficient conditions for diagnosability of permanent failures in single-failure scenarios, assuming $z_0 = X$.

**Theorem 2.1** Assume single-failure scenario and $z_0 = X$. A permanent failure $F_i$ is diagnosable if and only if

1. From every $x \in X_{F_i}$, there is (at least) one transition to another state in $X_{F_i}$ unless $\lambda(x)$ is $F_i$-indicative;

2. There is no cycle in $X_{F_i}$ consisting of states having the same output unless the output symbol is $F_i$-indicative;

3. There are no $F_i$-indeterminate cycles in the diagnoser. \hfill \qed

We need the following trivial lemma for the proof of Theorem 2.1.

**Lemma 2.2** Consider a path $z_0 \xrightarrow{y_1} z_1 \xrightarrow{y_2} z_2 \cdots \xrightarrow{y_n} z_n$ ($n \geq 1$) in the diagnoser. For any $x_n \in z_n$, there exist $x_i \in z_i$ ($1 \leq i \leq n - 1$) such that $x_i \Rightarrow x_{i+1}$ for $1 \leq i \leq n - 1$. \hfill \qed

**Proof of Theorem 2.1.** Conditions 1 and 2 guarantee that after $F_i$ occurs, new output symbol(s) will be generated and the output sequence can only terminate (if
it does) in an $F_i$-indicative symbol. Suppose either 1 or 2 does not hold. After $F_i$ occurs, the output sequence may end in a non-$F_i$-indicative output. If the diagnoser is initialized after this last symbol is generated, then it will not be able to detect and isolate the failure. Therefore, both 1 and 2 are necessary for diagnosability; from now on, assume they hold.

After the occurrence of $F_i$ and the generation of a new output symbol, the diagnoser state will be either $F_i$-certain or $F_i$-uncertain. If it is $F_i$-certain, then it will remain $F_i$-certain (because $F_i$ is permanent) and $F_i$ is diagnosed. If the diagnoser state is $F_i$-uncertain, then by condition 3 and the fact that the number of $F_i$-uncertain states is bounded, after the generation of a bounded number of output symbols, the diagnoser will reach an $F_i$-certain state (In general, the diagnoser gets trapped indefinitely in a cycle of $F_i$-uncertain states only if the cycle is $F_i$-indeterminate). Let $n_i$ denote the number of events that it takes the diagnoser to detect and isolate $F_i$. By condition 3, after the occurrence of the failure, the diagnoser can visit an $F_i$-uncertain state $z$ at most $|z \cap X_{F_i}|$ times. Hence

$$n_i \leq c_i \times p_i + p_i$$

where $c_i = \Sigma\{|z_k \cap X_{F_i}| \mid z_k \in Z \& z_k is F_i-uncertain\}$ and $p_i$ is the length of the longest path of faulty states having the same non-$F_i$-indicative output. By condition 2, $p_i \leq |X_{F_i}|$. Also $c_i \leq |X_{F_i}| \cdot |Z|$. Therefore

$$n_i \leq |X_{F_i}|(|X_{F_i}| \cdot |Z| + 1).$$

As a result, $F_i$ is diagnosable with $N_i = |X_{F_i}|(|X_{F_i}| \cdot |Z| + 1)$.

Conversely, if condition 3 is not true, then there exists an output sequence $y_1, y_2, \ldots, y_n$ that can take the diagnoser into a state $z_k$ belonging to an $F_i$-indeterminate cycle. Let $x_n \in z_n$ belong to an underlying faulty cycle in the RTS. By Lemma 2.2, there exist states $x_1, \ldots, x_{n-1}$ such that $x_i \Rightarrow x_{i+1}$ for $1 \leq i \leq n-1$. After reaching $x_n$, the RTS state may remain on the underlying faulty cycle causing the diagnoser to stay on the $F_i$-indeterminate cycle indefinitely. Therefore, there ex-
Figure 2.9: With $z_0 = \{0\}$, $F$ will be diagnosable.

ists a trajectory for the system leading to states in $X_F$, such that the corresponding output sequence throws the diagnoser into a cycle of $F_i$-uncertain states and keeps it there indefinitely. Hence, $F_i$ is not diagnosable.

Intuitively, $F_i$ would not be diagnosable if after it occurs, the system stops generating new output symbols (unless the last symbol is $F_i$-indicative). Also $F_i$ would be undiagnosable if after its occurrence, the system can generate a periodic output sequence that throws the diagnoser into a cycle of $F_i$-uncertain states.

**Remark 2.3** Let $D_1$ and $D_2$ be two diagnosers for a system, corresponding to different initial state estimates $z_0^1$ and $z_0^2$. If $z_0^1 \subseteq z_0^2$, then $z_k^1 \subseteq z_k^2$ for $k \geq 0$ ($z_k^1$ and $z_k^2$ are the state estimates of the diagnosers) because the diagnoser update law $\zeta(z_k, y_{k+1})$ is monotone in $z_k$ (i.e., $z_k^1 \subseteq z_k^2$ implies $\zeta(z_k^1, y_{k+1}) \subseteq \zeta(z_k^2, y_{k+1})$). Therefore in Theorem 2.1, if $z_0 \neq X$, then the set of conditions 1, 2 and 3 becomes only sufficient (not necessary) for diagnosability. Conditions 1 and 2 are not necessary in general. For example, consider the system given in Fig. 2.9. Here $X_N = \{0, 1, 2\}$ and $X_F = \{3\}$. Suppose $z_0 = \{0\}$. Condition 1 is not satisfied since $\beta$ is not $F$-indicative. The failure mode $F$, however, is diagnosable because if it occurs a $\beta$ will be generated immediately after $\alpha$ while in the normal mode, $\beta$ occurs only after $\gamma$. Condition 3, on the other hand, is necessary for any $z_0$. This follows from an argument similar to the one given for the necessity of condition 3 in the proof of Theorem 2.1.

**Remark 2.4** In control we are usually interested in diagnosis of permanent failures. But sometimes we may have to deal with nonpermanent failures too. For example, a valve may get stuck for a while and then return to normal operation. In these cases,
if the nonpermanent failure persists long enough, the diagnoser may finally diagnose it. If on the other hand, the system returns to normal operation after a short while, then the failure may go undetected. Of course the diagnoser will recognize that the system has returned to normal condition. If it is necessary to detect nonpermanent failures, then (in addition to estimating the system condition) one needs to detect the occurrence of failure events using either the method explained in Remark 2.1 or the event-based approach [29].

**Remark 2.5** Consider a valve which is closed. If it fails stuck-closed and no open commands are sent to the valve after the failure, the diagnoser will have no way of detecting the failure and according to Def. 2.4, the failure is not diagnosable. One may ignore the undiagnosability of the valve because the stuck-closed failure does not affect the performance of the system (The valve was closed prior to failure and no open commands has been issued afterwards). However, if the valve is a critical component of the system and, say, it is going to be used in another cycle of the operation of the plant and hence, the condition of the valve needs to be known, then the undiagnosability of the valve could be a problem that should be addressed.

In [29], Sampath et al. propose the notion of I-diagnosability to deal with the above problem. According to [29], the valve failure should be deemed undiagnosable (un-I-diagnosable to be specific) only if it cannot be diagnosed in a case where it is followed by an open command. Otherwise, the failure is considered I-diagnosable. This implies that if the valve fails stuck-closed while it is open and no commands are issued afterwards, the failure can still be considered (I-)diagnosable even if the diagnoser cannot detect and isolate the failure. This definition of diagnosability does not seem to be practically suitable.

We consider that this matter and similar cases involving the diagnosability of failures which may occur while the corresponding component is not ‘active’ should be analyzed on a case-by-case basis.

In the example of the heating system, $F$ is not diagnosable because the states 15 and 16 form a cycle which contradicts condition 2 in Theorem 2.1.
Example 2.2 - Pump and Valve [30]
Consider a system consisting of a pump, a valve and a flowmeter (Fig. 2.10). The valve may fail stuck-closed ($F_1$) or stuck-open ($F_2$). The pump is assumed reliable. The DES controller orders the valve open ($VE$), then turns on the pump ($PE$). After a while it shuts down the pump ($PD$) and closes the valve ($VD$). This cycle repeats. The finite-state automata describing the components of the system are shown in Fig. 2.11. The output of the flowmeter is either $f$ (flow) or $nf$ (no flow). The controller has 4 states. The current state of the controller is assumed to be known to the diagnoser. Therefore $Y \subseteq \{f, nf\} \times \{C1, C2, C3, C4\}$. The DES model of the system is the synchronous product of the components and is shown in Fig. 2.12. The RTS is given in Table 2.2 and the diagnoser in Fig. 2.13. Unlike [30], no prior information about the state and condition of the plant is assumed here, i.e., $z_0 = X$. 

27
Figure 2.12: Example 2.2. DES model of the pump and valve system.

<table>
<thead>
<tr>
<th>State</th>
<th>Output-adjacent states (output)</th>
<th>State</th>
<th>Output-adjacent states (output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2, 6, 10(C2, nf)</td>
<td>7</td>
<td>12(C4, nf)</td>
</tr>
<tr>
<td>2</td>
<td>3, 11(C3, f)/7(C3, nf)</td>
<td>8</td>
<td>9(C1, nf)</td>
</tr>
<tr>
<td>3</td>
<td>4, 12(C4, nf)/7(C3, nf)</td>
<td>9</td>
<td>6(C2, nf)</td>
</tr>
<tr>
<td>4</td>
<td>1, 5, 9(C1, nf)</td>
<td>10</td>
<td>7(C3, nf)</td>
</tr>
<tr>
<td>5</td>
<td>10(C2, nf)</td>
<td>11</td>
<td>8(C4, nf)</td>
</tr>
<tr>
<td>6</td>
<td>11(C3, f)</td>
<td>12</td>
<td>5(C1, nf)</td>
</tr>
</tbody>
</table>

Table 2.2: Example 2.2. Reachability transition system.
Figure 2.13: Example 2.2. Diagnoser.
The system does not have any deadlock states or cycles with constant output. Furthermore, there are no $F_1$-uncertain, hence, no $F_1$-indeterminate cycles in the diagnoser. Therefore, by Theorem 2.1, $F_1$ is diagnosable. The reader can verify that $F_1$ can be detected and isolated in at most 3 events. The failure $F_2$, on the other hand, is not diagnosable since $\{1, 5, 9\}, \{2, 6, 10\}, \{3, 11\}$ and $\{4, 12\}$ form an $F_2$-indeterminate cycle, with 9, 10, 11 and 12 as the underlying faulty cycle. Physically, if the valve fails stuck-closed when the pump is on, no flow will be registered by the flowmeter indicating failure $F_1$. In fact, $[C3, nf]$ is $F_1$-indicative. The stuck-open failure, however, does not produce a measurable effect at the output, i.e., the output sequence in $F_2$ mode is identical to that in the normal mode.

2.4 Model Reduction

The computational complexity of designing the diagnoser in the worst case is exponential in the number of system states. To mitigate this, in this section we introduce a model reduction scheme with polynomial time complexity. The equivalence relation used for model reduction is based on the solution of the relational coarsest partition (RCP) problem for the reachability transition system. We will show that the diagnoser built using the reduced RTS will be 'equivalent' to the original diagnoser in the following sense.

Definition 2.8 Two diagnosers for a system are equivalent if for any given output sequence, they produce identical sequences of estimates for the system's condition. □

Consider the RTS $\tilde{G} = (X, R, Y, \lambda)$. For every $x_1, x_2 \in X$, let $x_2 \in R(x_1)$ iff $(x_1, x_2) \in R$, i.e., $x_1 \Rightarrow x_2$. Let $\pi = \{B_1, \ldots, B_{|\pi|}\}$ be a partition of $X$, with $B_i$ denoting the blocks of $\pi$. The partition $\pi$ is said to be compatible with $R$ [10] if and only if whenever $x$ and $x'$ are in the same block $B_i$, then for any block $B_j$, $R(x) \cap B_j \neq \emptyset$ iff $R(x') \cap B_j \neq \emptyset$. Let $\Pi$ be the set of partitions compatible with $R$.

Suppose $\pi$ is compatible with $R$ and $\pi \leq \ker \lambda \land \ker \kappa$, where $\ker$ refers to the equivalence kernel of the corresponding map and $\land$ denotes the meet operation in
the lattice of equivalence relations [23]. If two states \(x\) and \(x'\) are in the same block of \(\ker \lambda \land \ker \kappa\), then the system has the same output and condition at these states. Moreover, the set of output sequences generated by the system, starting at either of these states, will be identical. Also, starting at any of these states and for any feasible output sequence generated, the sequence of condition estimates will be the same. This shows that the three statements \(x \in z, x' \in z\) and \(x, x' \in z\) contain the same information about the present and future estimates of the system's condition. Hence, for the purpose of estimating the system's condition, \(x\) and \(x'\) are equivalent.

Let \(P : X \to X/\pi\) be the canonical projection. For every \(x \in X\), \([x] := Px\) denotes the block \(x\) belongs to. Also for simplicity, instead of \(x \in P^{-1} \bar{x}_1\), we write \(x \in \bar{x}_1\). We define the reduced RTS \(\bar{G} = (\bar{X}, \bar{R}, Y, \bar{\lambda})\) (corresponding to \(\pi\) according to:

- \(\bar{X} = X/\pi:\)
- For all \(\bar{x}_1, \bar{x}_2 \in \bar{X} :\)
  \[(\bar{x}_1, \bar{x}_2) \in \bar{R} \iff (\forall x \in \bar{x}_1 \exists x' \in \bar{x}_2 : (x, x') \in R);\]
- For all \(\bar{x} \in \bar{X} : \bar{\lambda}(\bar{x}) = \lambda(x)\), for any \(x \in \bar{x}\).

Similarly we define \(\bar{\kappa} : \bar{X} \to \mathcal{K}\) according to \(\bar{\kappa}(\bar{x}) = \kappa(x)\) for any \(x \in \bar{x}\). Since \(\pi \leq \ker \lambda \land \ker \kappa, \bar{\lambda}\) and \(\bar{\kappa}\) are well-defined. We define the canonical projection of a subset \(z \subseteq X\) to be \(P(z) := \bigcup \{[x] | x \in z\}\).

We refer to the diagnoser designed based on the reduced RTS \(\bar{G}\) with \(\bar{z}_0 := Pz_0\) as its initial state estimate, as the high-level diagnoser (corresponding to \(\pi\)) and call it \(\bar{D}\).

**Theorem 2.3** The original diagnoser and the high-level diagnoser are equivalent.

**Proof.** We have to show that for every output sequence \((y_1, y_2, \cdots, y_j)\) and any initial state estimate \(z_0\), the diagnosers produce the same sequence of estimates for the condition of system, i.e., \(\kappa(z_k) = \bar{\kappa}(\bar{z}_k)\) for \(0 \leq k \leq j\) (\(\bar{z}_k\) is the state of \(\bar{D}\)).
Let $\zeta$ denote the transition (partial) function of $\overline{D}$. First we show that $Pz_k = \bar{z}_k$ (Fig. 2.14).

By definition $Pz_0 = \bar{z}_0$.

\[
Pz_1 = P(z_0 \cap \lambda^{-1}(\{y_1\})) = Pz_0 \cap P(\lambda^{-1}(\{y_1\})) = \bar{z}_0 \cap \lambda^{-1}(\{y_1\}) \quad \text{(because } \pi \leq \ker \lambda) = \bar{z}_1.
\]

Now we prove that for $k \geq 1$ if $Pz_k = \bar{z}_k$ then $Pz_{k+1} = \bar{z}_{k+1}$.

Let $x_{k+1}$ be an element of $z_{k+1}$, i.e., $x_{k+1} \in z_{k+1}$. Then there exists $x_k \in z_k$ such that $(x_k, x_{k+1}) \in R$. Hence $([x_k], [x_{k+1}]) \in \overline{R}$. From $Pz_k = \bar{z}_k$ it follows that $[x_k] \in \bar{z}_k$. Also $\lambda([x_{k+1}]) = \lambda(x_{k+1}) = y_{k+1}$. As a result $[x_{k+1}] \in \bar{z}_{k+1}$. This shows that $Pz_{k+1} \subseteq \bar{z}_{k+1}$.

Now let $\bar{x}_{k+1}$ be an element of $\bar{z}_{k+1}$, i.e., $\bar{x}_{k+1} \in \bar{z}_{k+1}$. Therefore $(\bar{x}_k, \bar{x}_{k+1}) \in \overline{R}$ for some $\bar{x}_k \in \bar{z}_k = Pz_k$. Hence there exist $x'_k \in \bar{x}_k \cap z_k$ and $x'_{k+1} \in \bar{x}_{k+1}$ such that $(x'_k, x'_{k+1}) \in R$. Also $\lambda(x'_{k+1}) = \lambda(\bar{x}_{k+1}) = y_{k+1}$. Thus $x'_{k+1} \in z_{k+1}$ and as a result $\bar{x}_{k+1} = Px'_{k+1} \in Pz_{k+1}$. This proves $\bar{z}_{k+1} \subseteq Pz_{k+1}$.

Therefore, by induction, $Pz_k = \bar{z}_k$ for $0 \leq k \leq j$. Finally $\kappa(\bar{z}_k) = \kappa(Pz_k) = \kappa(z_k)$ since $\pi \leq \ker \kappa$.

**Remark 2.6** Since $Pz_k = \bar{z}_k$ and $P$ is onto, the number of states of $\overline{D}$ is less than or equal to that of $D$. 

32
Remark 2.7 Obviously the coarser the partition $\pi$, the fewer states will the reduced RTS have. The set $\{\pi \in \Pi, \pi \leq \ker \lambda \land \ker \kappa\}$ is closed under $\lor$, the join operation in the lattice of equivalence relations, and therefore has a unique supremal element which is the coarsest partition compatible with $R$ and finer than $\ker \lambda \land \ker \kappa$. The problem of computing this supremal element is called the relational coarsest partition (RCP) problem. Iterative algorithms for solving the RCP problem exist in the literature (see [13] and references therein). The efficient algorithm of [24] solves the RCP problem in $\mathcal{O}(\log |X|)$ time ($|R|$ is the cardinality of the set $R$). This gives a $\mathcal{O}(|X|^2 \log |X|)$ time complexity for model reduction, since $|R| \leq |X|(|X| - 1)$.

Remark 2.8 Sometimes the output map $\lambda$ provides redundant information, i.e., fault diagnosis can be accomplished using a coarser output map. The use of a coarser output map, in general, results in further aggregation in model reduction, hence fewer states in the reduced RTS.

Remark 2.9 The high-level diagnoser produces an accurate estimate of the system's condition, with an accuracy proportional to the amount of information available in the reduced RTS iff the relation $P_z = \bar{z}$ holds. In the proof of Theorem 2.3, we showed that $P_z = \bar{z}$ holds for a partition $\pi \leq \ker \lambda \land \ker \kappa$ if $\pi$ is compatible with $R$. If $\pi$ is not compatible with $R$, then we can find two blocks of $\pi$, say $\bar{x}_1$ and $\bar{x}_2$, and states $x_1, x'_1 \in \bar{x}_1$ and $x_2 \in \bar{x}_2$ such that $(x_1, x_2) \in R$ but there is no $x \in \bar{x}_2$ such that $(x', x) \in R$. In general, if in this situation we let $(\bar{x}_1, \bar{x}_2) \in R$, then the high-level state estimates will become conservative, i.e., $P_z \subseteq \bar{z}$, and if we set $(\bar{x}_1, \bar{x}_2) \not\in R$, then the high-level state estimates will be risk-accepting, i.e., $\bar{z} \subseteq P_z$. Thus the use of compatible partitions is necessary for guaranteeing $P_z = \bar{z}$ and in this case, the coarsest partition compatible with $R$ and finer than $\ker \lambda \land \ker \kappa$, will be the optimal partition for model reduction, i.e., it will give the reduced RTS with minimum number of states.

Example 2.1 - Heating System (Cont'd)
Using Table 2.1, the reader can verify that the partition $\ker \lambda \land \ker \kappa = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \cdots, \{23, 24\}\}$ is compatible with the transition relation $R$ of the RTS.
Hence it is the solution to the RCP problem for this example. The reduced RTS is given in Table 2.3 where each pair of states $2i - 1$ and $2i$ $(1 \leq i \leq 12)$ in the original RTS is replaced with $i'$ in the reduced RTS. Therefore the number of states of the RTS has been reduced to half. The high-level diagnoser is identical to the original one (after replacing each pair of $2i - 1$ and $2i$ with $i'$).

The fact that the pair of states $2i$ and $2i - 1$ are equivalent and can be replaced by the single state $i'$, shows that incorporating the model of the load has not added any useful information to the heating system's model for the purpose of fault diagnosis and therefore, the load model can be removed. This interesting point could be of use in decentralized fault diagnosis. Think of the load as subsystem 1 and of the rest of the heating system as subsystem 2. Our model reduction scheme has shown that in the absence of sensor readings from subsystem 1 (which is typical in a decentralized fault detection scheme), the model of subsystem 2 (the local model) has the same amount of information for fault diagnosis as the combined model of the subsystems (the centralized model). Thus the design of diagnoser for subsystem 2 can be done based on the local model only.

In order to verify condition 3 of Theorem 2.1, one needs to build the diagnoser. We will show in the following that condition 3 can be replaced with a similar condition in terms of the $F_i$-indeterminate cycles of the high-level diagnoser. Therefore there will be no need to build the low-level diagnoser for checking diagnosability.

<table>
<thead>
<tr>
<th>State</th>
<th>Output-adjacent states (output)</th>
<th>State</th>
<th>Output-adjacent states (output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1'</td>
<td>2', 8' (le)</td>
<td>7'</td>
<td>8' (le)</td>
</tr>
<tr>
<td>2'</td>
<td>3' (be)</td>
<td>8'</td>
<td>-</td>
</tr>
<tr>
<td>3'</td>
<td>4' (ae) / 8' (le)</td>
<td>9'</td>
<td>8' (le)</td>
</tr>
<tr>
<td>4'</td>
<td>5', 11' (ad)</td>
<td>10'</td>
<td>11' (ad)</td>
</tr>
<tr>
<td>5'</td>
<td>6', 12' (bd)</td>
<td>11'</td>
<td>12' (bd)</td>
</tr>
<tr>
<td>6'</td>
<td>3', 9' (be)</td>
<td>12'</td>
<td>9' (be)</td>
</tr>
</tbody>
</table>

Table 2.3: Example 2.1 Reduced reachability transition system.
The low-level and high-level diagnosers produce identical estimates for the system's condition. Therefore, a failure mode $F_i$ is diagnosable if and only if it can be detected and isolated by the high-level diagnoser (i.e., the high-level diagnoser reaches an $F_i$-certain state) after the occurrence of a bounded number of events, following the failure and initialization of the high-level diagnoser. As with the low-level diagnoser, a state $\bar{z}$ of the high-level diagnoser is called $F_i$-certain if the estimate $\bar{\kappa}(\bar{z})$ indicates that the failure $F_i$ has occurred. In a single-failure scenario, $\bar{z}$ is $F_i$-certain iff $\bar{\kappa}(\bar{z}) = \{F_i\}$. A state $\bar{z}$ of the high-level diagnoser is called $F_i$-uncertain if $\bar{z}$ is not $F_i$-certain but $\bar{\kappa}(\bar{z})$ indicates that the failure $F_i$ might have occurred, i.e., occurrence of $F_i$ is consistent with the observed output sequence. In a single-failure scenario, $\bar{z}$ is $F_i$-uncertain iff $F_i \in \bar{\kappa}(\bar{z})$ but $\bar{\kappa}(\bar{z}) \neq \{F_i\}$.

**Definition 2.9** Suppose $\bar{z}^1, \ldots, \bar{z}^m$ is a cycle of $F_i$-uncertain states of the high-level diagnoser. The cycle is called $F_i$-indeterminate (wrt the reduced RTS) if there exist $l \geq 1$ and $\bar{x}_1^1, \bar{x}_2^1, \ldots, \bar{x}_l^1 \in \bar{z}^1$, for all $1 \leq j \leq m$ such that $\bar{x}_j^k \in \bar{X}_{F_i}$ for all $1 \leq j \leq m$, $1 \leq k \leq l$ and $\bar{x}_1^1, \bar{x}_1^2, \ldots, \bar{x}_1^m, \bar{x}_2^1, \ldots, \bar{x}_2^m, \ldots, \bar{x}_l^1, \ldots, \bar{x}_l^m$ form a cycle in the reduced RTS. The cycle in the reduced RTS is called an underlying faulty cycle of the $F_i$-indeterminate cycle.

We need the following two lemmas which relate the cycles in the RTS to those in the reduced RTS.

**Lemma 2.4** Let $x_1, \ldots, x_m$ be a cycle in the RTS. Then $[x_1], \ldots, [x_m]$ is a cycle in the reduced RTS.

**Proof.** Follows from: $(x, x') \in R$ iff $(\bar{x}, \bar{x}') \in \bar{R}$, for all $x, x' \in X$.

**Lemma 2.5** Let $\bar{x}_1, \ldots, \bar{x}_m$ be a cycle in the reduced RTS. Starting at any $x_1 \in \bar{x}_1$, we can construct a path in the RTS such that either the path is a cycle or it leads to a cycle in the RTS, and the projection of the cycle (and the path) in the reduced RTS is the cycle $\bar{x}_1, \ldots, \bar{x}_m$.

**Proof.** Let $x_1 \in \bar{x}_1$. $(\bar{x}_1, \bar{x}_2) \in \bar{R}$; therefore there exists $x_2 \in \bar{x}_2$ and $(x_1, x_2) \in R$. Continuing in this way, we can build a path $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \cdots \rightarrow x_m \rightarrow \cdots$ in
the RTS, where \( x_i \in \bar{x}_j \) with \( i = km + j \) for some \( k \geq 0 \) and \( 1 \leq j \leq m \). Since the number of elements of \( \bar{x}_i \) is finite, there exist \( k_1 \) and \( k_2 \) with \( 0 \leq k_1 < k_2 \leq |\bar{x}_1| \) such that \( x_{k_1m+1} = x_{k_2m+1} \). This means that the element \( x_{k_1m+1} \) appears in the path for a second time. Therefore \( x_{k_1m+1}, \ldots, x_{k_2m} \) form a cycle in the RTS. Obviously the projection of this cycle (and that of the path) into \( \bar{X} \) will be the cycle \( \bar{x}_1, \ldots, \bar{x}_m \). \( \square \)

**Theorem 2.6** Assume single-failure scenario and \( z_0 = X \). A permanent failure \( F_i \) is diagnosable if and only if

1. From every \( x \in X_{F_i} \), there is (at least) one transition to another state in \( X_{F_i} \) unless \( \lambda(x) \) is \( F_i \)-indicative;

2. There is no cycle in \( X_{F_i} \) consisting of states having the same output unless the output symbol is \( F_i \)-indicative;

3. There are no \( F_i \)-indeterminate cycles in the high-level diagnoser.

**Proof.** It was shown in the proof of Theorem 2.1 that conditions 1 and 2 are necessary. So from now on we assume they hold.

Suppose condition 3 holds. Then there will be no \( F_i \)-indeterminate cycles in the low-level diagnoser too. This is because if \( z^1, \ldots, z^m \) is an \( F_i \)-indeterminate cycle of the low-level diagnoser with \( x^1, \ldots, x^l \) as an underlying faulty cycle in the RTS, then by Lemma 2.4, \( Pz^1, \ldots, Pz^m \) and \( [x^1], \ldots, [x^l] \) will be an \( F_i \)-indeterminate cycle of the high-level diagnoser and an underlying faulty cycle in the reduced RTS, respectively. (Note that \( z \) is \( F_i \)-uncertain if and only if \( Pz \) is \( F_i \)-uncertain since \( \kappa(z) = \kappa(Pz) \)). It follows from Theorem 2.1 that \( F_i \) is diagnosable.

Conversely, assume condition 3 is not true. Then there exists an output sequence \( y_1, y_2, \ldots, y_k \) that can lead the high-level diagnoser into a state \( \bar{z}_k = Pz_k \) belonging to an \( F_i \)-indeterminate cycle (\( z_k \) is the state of the low-level diagnoser after observing \( y_1, y_2, \ldots, y_k \)). There exists \( \bar{x}_k \in \bar{x}_k \) belonging to an underlying faulty cycle in the reduced RTS. Let \( x_k \in \bar{x}_k \cap z_k \). Note that \( x_k \in X_{F_i} \) since \( \bar{x}_k \in \bar{X}_{F_i} \). Since \( x_k \in z_k \), there exists a sequence of states of the RTS \( x_j \in z_j (1 \leq j \leq k) \) such that \( x_i \rightarrow x_{i+1} \) for \( 1 \leq i \leq k - 1 \). Using Lemma 2.5, we can find a path \( \{x_j\}_{j \geq k} \) in the RTS whose
projection into $\overline{X}$ will be the underlying faulty cycle that $\bar{x}_k$ belongs to. Note that $x_j \in X_{F_i}$ for $j \geq k$. As a result, using the path $\{x_i\}_{i \geq 1}$ in the RTS, we can construct a trajectory for the system that leads to states in $X_{F_i}$ and the corresponding output sequence of this trajectory takes the high-level (and low-level) diagnoser into a cycle of $F_i$-uncertain states and keeps it there. Therefore, $F_i$ is not diagnosable. 

**Remark 2.10** Since the high-level and low-level diagnosers produce identical condition estimates, the upper bounds obtained for $n_i$ in the proof of Theorem 2.1, namely, $c_i \times p_i + p_i$ and $|X_{F_i}|(|X_{F_i}| |Z| + 1)$, also hold for $\bar{n}_i$, the number of events it takes the high-level diagnoser to detect and isolate a diagnosable failure $F_i$. The following upper bounds, however, can be obtained directly as well:

$$\bar{n}_i \leq \bar{c}_i \times p_i + p_i$$
$$\bar{n}_i \leq |X_{F_i}|(|X_{F_i}| |\bar{Z}| + 1)$$

with $p_i$ as before and

$$\bar{c}_i = \Sigma\{|P^{-1} \bar{z}_k \cap X_{F_i}| \mid \bar{z}_k \text{is } F_i\text{-uncertain}\}$$
$$\bar{Z} = \text{The state set of the high-level diagnoser} - \{\bar{z}_0\}.$$  

**2.5 Simultaneous Failures**

In this section, results of the previous sections on single-failure scenario will be extended to the case of simultaneous failures. We assume that there are only two failure modes. Generalization to arbitrary numbers of failures follows similarly.

Let us assume that there are two failure modes $F_1$ and $F_2$. There are three failure scenarios: $F_1$, $F_2$ and $F_{12}$, with $F_{12}$ denoting simultaneous occurrence of $F_1$ and $F_2$. Hence the condition set is $\mathcal{K} = \{N, F_1, F_2, F_{12}\}$. The state set is partitioned according to condition: $X = X_N \cup X_{F_1} \cup X_{F_2} \cup X_{F_{12}}$. The condition map $\kappa : X \rightarrow \mathcal{K}$
is defined as follows: \( \kappa(x) = N \) if \( x \in X_N \), \( \kappa(x) = X_{F_i} \) if \( x \in X_{F_i} \) \((i \in \{1, 2\})\), and \( \kappa(x) = F_{12} \) if \( x \in X_{F_{12}} \). The condition map is extended to the subsets of \( X \) according to \( \kappa(z) = \{ \kappa(x) \mid x \in z \} \) for all \( z \subseteq X \). The RTS and the diagnoser are defined and constructed as in Section 2.2.

In the rest of this section we consider the diagnosability of the failure mode \( F_1 \).

It follows from Def. 2.2 that in the case studied in this section, a state \( z \) of the diagnoser is \( F_1 \)-certain iff \( \kappa(z) = \{ F_1 \} \) or \( \kappa(z) = \{ F_{12} \} \) or \( \kappa(z) = \{ F_1, F_{12} \} \). Also \( z \) is \( F_1 \)-uncertain iff \( z \) is not \( F_1 \)-certain and \( \kappa(z) \cap \{ F_1, F_{12} \} \neq \emptyset \).

The failure modes \( F_1 \) and \( F_2 \) are assumed permanent. This means that \( X_N \) is not reachable from \( X_{F_1} \cup X_{F_2} \cup X_{F_{12}} \); neither \( X_{F_1} \) nor \( X_{F_2} \) is reachable from \( X_{F_{12}} \). There may be transitions directly from states in \( X_N \) to states in \( X_{F_{12}} \) (i.e., failure events \( F_1 \) and \( F_2 \) may happen at the same time) (Fig. 2.15).

The definition of diagnosability is the same as in Def. 2.4. The definition of \( F_1 \)-indicative output symbols remains the same (Def. 2.5). In our case, an output symbol \( y \) is \( F_1 \)-indicative iff \( \lambda^{-1}(\{y\}) \subseteq X_{F_1} \cup X_{F_{12}} \). Necessary and sufficient conditions for diagnosability of \( F_1 \) which will be given in Theorem 2.7 are the same as those in the single-failure scenario except that \( X_{F_1} \) is replaced with \( X_{F_1} \cup X_{F_{12}} \). This is because after the occurrence of \( F_1 \), the state \( x \) will be in \( X_{F_1} \cup X_{F_{12}} \). Note that since the failures are assumed permanent, there are no transitions from the states of \( X_{F_{12}} \) to those of \( X_{F_1} \). Also any cycle in \( X_{F_1} \cup X_{F_{12}} \) is entirely in either \( X_{F_1} \) or \( X_{F_{12}} \).
$F_1$-indeterminate cycles are defined as follows.

**Definition 2.10** Suppose $z_1, \ldots, z^m$ is a cycle of $F_1$-uncertain states of the diagnoser. The cycle is called **$F_1$-indeterminate** if there exist $l \geq 1$ and $x_1^l, x_2^l, \ldots, x_l^l \in z^l$ for all $1 \leq j \leq m$ such that $\{x_k^j | 1 \leq j \leq m, 1 \leq k \leq l\} \subseteq X_{F_1}$ or $\{x_k^j | 1 \leq j \leq m, 1 \leq k \leq l\} \subseteq X_{F_{12}}$, and also $x_1^1, x_1^2, \ldots, x_1^m, x_2^1, \ldots, x_2^m, \ldots, x_l^1, \ldots, x_l^m$ form a cycle in the RTS. The RTS cycle is called an **underlying faulty cycle** of the $F_1$-indeterminate cycle. \hfill \Box

**Theorem 2.7** Suppose $z_0 = X$. The permanent failure $F_1$ is diagnosable if and only if

1. From every $x \in X_{F_1} \cup X_{F_{12}}$, there is (at least) one transition to another state in $X_{F_1} \cup X_{F_{12}}$ unless $\lambda(x)$ is $F_1$-indicative;

2. There is no cycle in $X_{F_1}$ or in $X_{F_{12}}$ consisting of states having the same output unless the output symbol is $F_1$-indicative;

3. There are no $F_1$-indeterminate cycles in the diagnoser.

**Proof.** Similar to the proof of Theorem 2.1. \hfill \Box

The reduced RTS and high-level diagnoser can be constructed in a way similar to that described in Section 2.4. The low-level and high-level state estimates will still be related by: $P z_k = \tilde{z}_k$. The results of Section 2.4 on diagnosability can be similarly extended to the case studied in this section. We briefly point them out here.

A state of the high-level diagnoser $\tilde{z}$ is $F_1$-certain if $\tilde{\kappa}(\tilde{z}) = \{F_1\}$ or $\tilde{\kappa}(\tilde{z}) = \{F_{12}\}$ or $\tilde{\kappa}(\tilde{z}) = \{F_1, F_{12}\}$. $\tilde{z}$ is $F_1$-uncertain if it is not $F_1$-certain and $\tilde{\kappa}(\tilde{z}) \cap \{F_1, F_{12}\} \neq \emptyset$. The failure mode $F_1$ is diagnosable if and only if it can be detected and isolated by the high-level diagnoser (i.e., the high-level diagnoser enters an $F_1$-certain state) after the occurrence of a bounded number of events, following the failure and initialization of the high-level diagnoser.

The diagnosability conditions, to be given later, are the same as those in the single-failure scenario except that $X_{F_1}$ and $\overline{X}_{F_1}$ are replaced by $X_{F_1} \cup X_{F_{12}}$ and $\overline{X}_{F_1} \cup \overline{X}_{F_{12}}$, respectively.
respectively. This is simply because after the occurrence of $F_1$, the state $x$ will be in $X_{F_1} \cup X_{F_{12}}$. Note that since $F_1$ and $F_2$ are assumed permanent, in the reduced RTS there are no transitions from states of $X_{F_{12}}$ to those of $X_{F_1}$ (or $X_{F_2}$ for that matter). Also any cycle in $X_{F_1} \cup X_{F_{12}}$ will entirely be either in $X_{F_1}$ or in $X_{F_{12}}$. The definition of $F_1$-indeterminate cycle (with respect to the reduced RTS) is the same as Def. 2.7 (with $i = 1$) except that $z^i$, $x^i_k$, $X_{F_1}$ and $X_{F_{12}}$ are replaced with $z^i$, $x^j_k$, $X_{F_1}$ and $X_{F_{12}}$, respectively and the underlying cycle will be in the reduced RTS.

**Theorem 2.8** Suppose $z_0 = X$. The permanent failure $F_1$ is diagnosable if and only if

1. From every $x \in X_{F_1} \cup X_{F_{12}}$, there is (at least) one transition to another state in $X_{F_1} \cup X_{F_{12}}$ unless $\lambda(x)$ is $F_1$-indicative;

2. There is no cycle in $X_{F_1}$ or in $X_{F_{12}}$ consisting of states having the same output unless the output symbol is $F_1$-indicative;

3. There are no $F_1$-indeterminate cycles in the high-level diagnoser.

**Proof.** Similar to the proof of Theorem 2.6. \qed

**Remark 2.11** In the definition of diagnosability, a failure mode $F_i$ is called diagnosable if after the occurrence of the failure and initialization of the diagnoser, and occurrence of a bounded number of events, the diagnoser enters an $F_i$-certain state. This finite set of events may include failure events. One may argue that the detection of $F_i$ should not depend on the occurrence of another failure (or perhaps another occurrence of $F_i$). But we would like to point out that in a lot of cases, failures that may happen simultaneously are not independent and in fact, are related. Failures having a common cause and failures involved in a domino effect (where one failure causes another failure) are two such cases. In these situations, the definition of diagnosability given in this thesis seems reasonable. \qed

**Example 2.3 - Pump and Valve** [30]
Let us go back to the pump and valve system of Example 2.2 (Section 2.3).
time suppose the valve can only fail stuck-closed (failure mode $F_1$). Also assume that the pump can fail ON (failure mode $F_2$) and that the pump failure and valve failure may occur simultaneously. The DES model of the system are given in Fig. 2.16. We assume that a pressure sensor on the pump detects pressure and generates one of the two output symbols $pp$ (positive pressure) or $np$ (no pressure). In this system, $Y = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_3, \gamma_1, \gamma_2, \gamma_4, \delta_2, \delta_4\}$, where

\[
\begin{align*}
\alpha_1 &= (C1, nf, np), & \alpha_2 &= (C2, nf, np), \\
\alpha_3 &= (C3, f, pp), & \alpha_4 &= (C4, nf, np), \\
\beta_3 &= (C3, nf, pp), & \gamma_1 &= (C1, nf, pp), \\
\gamma_2 &= (C2, f, pp), & \gamma_4 &= (C4, f, pp), \\
\delta_2 &= (C2, nf, pp), & \delta_4 &= (C4, nf, pp).
\end{align*}
\]

The RTS and diagnoser are given in Tables 2.4 and 2.5. The reader can easily verify that conditions 1 and 2 of Theorem 2.7 are satisfied for both of the failure modes. Furthermore, there are no $F_1$-uncertain or $F_2$-uncertain cycles, hence neither $F_1$ nor $F_2$-indeterminate cycles, in the diagnoser and therefore, condition 3 holds as well. As a result, both $F_1$ and $F_2$ are diagnosable.

Intuitively, one can see that if the valve fails stuck-closed while the pump is ON (or failed ON), there will be no flow while positive pressure is detected (output symbol
<table>
<thead>
<tr>
<th>State</th>
<th>Output-adjacent states (output)</th>
<th>State</th>
<th>Output-adjacent states (output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,6 (α2) / 9,13 (γ1)</td>
<td>9</td>
<td>10 (γ2) / 14 (δ2)</td>
</tr>
<tr>
<td>2</td>
<td>3 (α3) / 7 (β4) / 10 (γ2) / 14 (δ2)</td>
<td>10</td>
<td>11 (α3) / 15 (β3)</td>
</tr>
<tr>
<td>3</td>
<td>4 (α4) / 7,15 (β3) / 12 (γ4)</td>
<td>11</td>
<td>12 (γ4) / 15 (β3)</td>
</tr>
<tr>
<td>4</td>
<td>1,5 (α1) / 12 (γ4) / 16 (δ4)</td>
<td>12</td>
<td>9 (γ1) / 16 (δ4)</td>
</tr>
<tr>
<td>5</td>
<td>6 (α2) / 13 (γ1)</td>
<td>13</td>
<td>14 (δ2)</td>
</tr>
<tr>
<td>6</td>
<td>7 (β3) / 14 (δ2)</td>
<td>14</td>
<td>15 (β3)</td>
</tr>
<tr>
<td>7</td>
<td>8 (α4) / 16 (δ4)</td>
<td>15</td>
<td>16 (δ4)</td>
</tr>
<tr>
<td>8</td>
<td>5 (α1) / 16 (δ4)</td>
<td>16</td>
<td>13 (γ1)</td>
</tr>
</tbody>
</table>

Table 2.4: Example 2.3. Reachability transition system.

<table>
<thead>
<tr>
<th>Diagnoser state</th>
<th>System output</th>
<th>Condition</th>
<th>Diagnoser next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>z₀ = X</td>
<td>–</td>
<td>K</td>
<td>{1, 5}, {2, 6}, {3, 11}, {4, 8}, {7, 15}, {9, 13}, {10}, {12}, {14}, {16}</td>
</tr>
<tr>
<td>{1, 5}</td>
<td>α₁</td>
<td>N, F₁</td>
<td>{2, 6}, {9, 13}</td>
</tr>
<tr>
<td>{2, 6}</td>
<td>α₂</td>
<td>N, F₁</td>
<td>{3}, {7}, {10}, {14}</td>
</tr>
<tr>
<td>{3, 11}</td>
<td>α₃</td>
<td>N, F₂</td>
<td>{4}, {7, 15}, {12}</td>
</tr>
<tr>
<td>{4, 8}</td>
<td>α₄</td>
<td>N, F₁</td>
<td>{1, 5}, {12}, {16}</td>
</tr>
<tr>
<td>{7, 15}</td>
<td>β₃</td>
<td>F₁, F₁₂</td>
<td>{8}, {16}</td>
</tr>
<tr>
<td>{9, 13}</td>
<td>γ₁</td>
<td>F₂, F₁₂</td>
<td>{10}, {14}</td>
</tr>
<tr>
<td>{3}</td>
<td>α₃</td>
<td>N</td>
<td>{4}, {7, 15}, {12}</td>
</tr>
<tr>
<td>{4}</td>
<td>α₄</td>
<td>N</td>
<td>{1, 5}, {12}, {16}</td>
</tr>
<tr>
<td>{5}</td>
<td>α₁</td>
<td>F₁</td>
<td>{6}, {13}</td>
</tr>
<tr>
<td>{6}</td>
<td>α₂</td>
<td>F₁</td>
<td>{7}, {14}</td>
</tr>
<tr>
<td>{7}</td>
<td>β₃</td>
<td>F₁</td>
<td>{8}, {16}</td>
</tr>
<tr>
<td>{8}</td>
<td>α₄</td>
<td>F₁</td>
<td>{5}, {16}</td>
</tr>
<tr>
<td>{9}</td>
<td>γ₁</td>
<td>F₂</td>
<td>{10}, {14}</td>
</tr>
<tr>
<td>{10}</td>
<td>γ₂</td>
<td>F₂</td>
<td>{11}, {15}</td>
</tr>
<tr>
<td>{11}</td>
<td>α₃</td>
<td>F₂</td>
<td>{12}, {15}</td>
</tr>
<tr>
<td>{12}</td>
<td>γ₄</td>
<td>F₂</td>
<td>{9}, {16}</td>
</tr>
<tr>
<td>{13}</td>
<td>γ₁</td>
<td>F₁₂</td>
<td>{14}</td>
</tr>
<tr>
<td>{14}</td>
<td>δ₂</td>
<td>F₁₂</td>
<td>{15}</td>
</tr>
<tr>
<td>{15}</td>
<td>β₃</td>
<td>F₁₂</td>
<td>{16}</td>
</tr>
<tr>
<td>{16}</td>
<td>δ₄</td>
<td>F₁₂</td>
<td>{13}</td>
</tr>
</tbody>
</table>

Table 2.5: Example 2.3. Diagnoser.
$\beta_3$); this indicates the existence of the failure $F_1$. Also if the pump fails ON at the start of cycle when the controller is at the state $C_1$, positive pressure will be detected (output symbol $\gamma_1$) indicating the presence of the failure $F_2$. Note that $\beta_3$ and $\gamma_1$ are $F_1$ and $F_2$ indicative, respectively.
Chapter 3

Fault Diagnosis in Timed Discrete–Event Systems

In Chapter 2, changes in the output sequence resulting from failures were used for diagnosing failures in finite-state Moore automata. In many cases, however, a failure changes only the timing of the output sequence, rather than the sequence itself. In other cases, as a result of failure, the system stops generating new output symbols (e.g., when the temperature is low in the heating system of Example 2.1). In these situations, timing information can be used to increase the accuracy and speed of fault diagnosis. This improvement, in general, comes at the expense of additional computational complexity.

In this chapter, we extend our framework to timed discrete–event systems (TDES). In Section 3.1, we discuss failure modelling. Section 3.2 presents an extended version of the diagnoser of Chapter 2 and explains how it can be used in fault diagnosis in TDES. In Section 3.3 diagnosability in a finite time interval is studied, the concept of time–diagnosability in our framework is introduced, and necessary and sufficient conditions for time–diagnosability are obtained.
3.1 Plant Model

In this chapter, we assume that the plant under control can be modelled as a timed discrete–event system (TDES) [3]. A TDES is a finite–state automaton containing in its event set an event which is the tick of a global clock. The tick is assumed to be observable. The TDES model can be used to describe the sequence of events occurring in the system with respect to the ticks of the global clock.

Formally, it is assumed that the plant under control can be described by the finite–state Moore automaton $G_{\tau} = (Q_{\tau}, \Sigma_{\tau}, \delta_{\tau}, q_{0}, Y_{\tau}, \lambda_{\tau})$, where $Q_{\tau}$, $\Sigma_{\tau}$ and $Y_{\tau}$ are the finite state, event and output sets; $q_{0}$ is the initial state, $\delta_{\tau} : Q_{\tau} \times \Sigma_{\tau} \rightarrow 2^{Q_{\tau}}$ the transition function and $\lambda_{\tau} : Q_{\tau} \rightarrow Y_{\tau}$ the output map. As noted above, one of the events in $\Sigma_{\tau}$ is the tick of the global clock, and is denoted here by $\tau$. It is assumed that $Q_{\tau}$ is reachable from $q_{0}$.

Note that $G_{\tau}$ (or any other TDES for that matter) cannot have deadlock states; i.e., for any $q \in Q_{\tau}$, $\delta_{\tau}(q, \sigma) \neq \emptyset$, for some $\sigma \in \Sigma_{\tau}$. This is because at least the tick event happens periodically, and therefore at each state $q \in Q_{\tau}$, even if none of the non–tick events is enabled, the tick event will be enabled ($\delta_{\tau}(q, \tau) \neq \emptyset$), and will occur at the due time.

It is assumed that $G_{\tau}$ is activity–loop–free [3], which simply means that there are no cycles of non–tick transitions in $G_{\tau}$; i.e., for every $n \geq 1$ and $q_{1}, \ldots, q_{n+1} \in Q_{\tau}$ and $\sigma_{1}, \ldots, \sigma_{n} \in \Sigma_{\tau} - \{\tau\}$ such that $q_{i+1} \in \delta(q_{i}, \sigma_{i})$ $(i \in \{1, \ldots, n\})$, we must have $q_{1} \neq q_{n+1}$. This means that only a finite number of non–tick events may occur between any two consecutive clock ticks.

The TDES $G_{\tau}$ describes the behaviour of the system in both normal and faulty situations. Suppose there are $p$ failure modes $F_{1}, \ldots, F_{p}$. Our discussion in this chapter will be limited to single–failure scenarios; the results can be generalized to the case of simultaneous failures as in Chapter 2.

It is assumed that the state set $Q_{\tau}$ can be partitioned according to the condition of the system: $Q = Q_{\tau,N} \cup Q_{\tau,F_{1}} \cup \cdots \cup Q_{\tau,F_{p}}$. With $\mathcal{K} = \{N, F_{1}, \ldots, F_{p}\}$ as the condition set, the condition map of $G_{\tau}$, $\kappa_{\tau} : Q_{\tau} \rightarrow \mathcal{K}$, is defined such that for every $q \in Q_{\tau}$,
\( \kappa_r(q) = N \) if \( q \in Q_{r,N} \), and \( \kappa_r(q) = F_i \) if \( q \in Q_{r,F_i} \) \((i \in \{1, \ldots, p\})\). Note that in our framework, the occurrences of failures are modelled by failure events; therefore in any transition \( q_1 \xrightarrow{\sigma} q_2 \), \( \kappa_r(q_1) \neq \kappa_r(q_2) \) only if \( \sigma \) is a failure event. Thus, if \( \sigma = \tau \), then \( \kappa_r(q_1) = \kappa_r(q_2) \).

Timed discrete-event systems are used in [3] in the supervisory control problem. There, the TDES models are obtained from activity transition graphs (ATG). ATG models provide a compact way of describing discrete-event processes. ATG models for complex systems can be systematically composed from the ATG models of the corresponding subsystems using a useful synchronous product [3]. Every ATG model can be transferred into a TDES (as defined in this work) but not vice versa; i.e., not every TDES represents an ATG model.

As mentioned before, ATG models are convenient for describing discrete-event processes and we will use them later in this section in an example. In the following, we describe ATG models and explain how TDES models can be obtained from them. For more details about ATG models, the reader can refer to [3].

Let us assume that the plant under control can be modelled with the six-tuple \( G_{\text{act}} = (A, \Sigma_{\text{act}}, \delta_{\text{act}}, a_0, Y_{\text{act}}, \lambda_{\text{act}}) \), which we call the activity transition system. \( A \), \( \Sigma_{\text{act}} \) and \( Y_{\text{act}} \) are the finite activity, event and output sets; \( a_0 \) is the initial activity, \( \delta_{\text{act}} : A \times \Sigma_{\text{act}} \rightarrow 2^A \) the activity transition function and \( \lambda_{\text{act}} : A \rightarrow Y_{\text{act}} \) the activity output map. This model is similar to that in [3] except that it is nondeterministic and includes an output map. The transition graph of \( G_{\text{act}} \) is called the activity transition graph (ATG).

Let \( a \) denote the activity of the system. Initially, \( a = a_0 \). For a given activity \( a \), an event \( \sigma \in \Sigma_{\text{act}} \) is enabled (resp. disabled) if \( \delta(a, \sigma) \neq \emptyset \) (resp. \( \delta(a, \sigma) = \emptyset \)). Each activity corresponds to a ‘mode’ or ‘phase’ of operation of the system in which certain events are enabled or remain enabled or are disabled. Each event \( \sigma \in \Sigma_{\text{act}} \) has a lower time bound \( l_\sigma \in \mathbb{N} \) and an upper time bound \( u_\sigma \in \mathbb{N} \cup \{\infty\} \), with \( l_\sigma \leq u_\sigma \) \((\mathbb{N} = \{0, 1, 2, \ldots\})\).

An event \( \sigma \) is called prospective if \( u_\sigma \in \mathbb{N} \) and remote if \( u_\sigma = \infty \). A prospective event \( \sigma \) with time bounds \( l_\sigma \) and \( u_\sigma \) cannot occur before \( l_\sigma \) ticks of the clock but if it
remains enabled, it will occur before the \((u_\sigma + 1)\)-th tick (following its enablement) unless it is preempted by another event in \(\Sigma_{act}\). If \(\sigma\) is remote with a lower time bound of \(l_\sigma\), then it will not occur before \(l_\sigma\) clock ticks but if it remains enabled, it may occur any time after the \(l_\sigma\)-th tick. The time bounds \(l_\sigma\) and \(u_\sigma\) typically would represent delays due to process dynamics or measurement.

To keep track of the status of each event, let us bring in the timers \(t_\sigma\) \((\sigma \in \Sigma_{act})\). Initially, by definition, the timers assume their default values given by

\[
t_{\sigma_0} = \begin{cases} u_\sigma & \text{\(\sigma\) prospective,} \\ l_\sigma & \text{\(\sigma\) remote.} \end{cases}
\]

If the system is in \(a \in A\) and \(\sigma\) is enabled (i.e., \(\delta(a, \sigma) \neq \emptyset\)), then \(t_\sigma\) will be decremented by one after each clock tick until \(t_\sigma = 0\). After this, \(t_\sigma\) will remain at 0. The timer \(t_\sigma\) will be reset to its default value if either \(\sigma\) occurs or \(\sigma\) is disabled. Therefore, a prospective event \(\sigma\) can happen only when \(0 \leq t_\sigma \leq u_\sigma - l_\sigma\); if \(t_\sigma\) becomes 0, then \(\sigma\) will occur before the next tick unless it is preempted by another event in \(\Sigma_{act}\). Also, a remote event \(\sigma\) may occur only when \(t_\sigma = 0\).

Define timer intervals \(T_\sigma \subseteq 2^N\) according to

\[
T_\sigma := \begin{cases} [0, u_\sigma] & \text{\(\sigma\) prospective,} \\ [0, l_\sigma] & \text{\(\sigma\) remote.} \end{cases}
\]

In this chapter, for \(j, k \in \mathbb{N}\), \([j, k] := \{i \in \mathbb{N} \mid j \leq i \leq k\}\). Obviously, \(t_\sigma \in T_\sigma\) always, for all \(\sigma \in \Sigma_{act}\).

The state of \(G_{act}\) is given by the current activity \(a\) and the timers \(t_\sigma\) \((\sigma \in \Sigma_{act})\). Formally, the evolution of the state of \(G_{act}\) is described by the timed discrete–event system \(G_\tau = (Q_\tau, \Sigma_\tau, \delta_\tau, q_0, Y_\tau, \lambda_\tau)\), where \(Q_\tau \subseteq A \times \prod\{T_\sigma \mid \sigma \in \Sigma_{act}\}\), \(\Sigma_\tau = \Sigma_{act} \cup \{\tau\}\) and \(Y_\tau = Y_{act}\) are the finite state, event and output sets. Note that the event set \(\Sigma_\tau\) includes the clock tick. At \(q_0\), the initial state, \(a = a_0\) and all of the
timers assume their default values. \( \lambda_r : Q_r \to Y_r \) is the output map with

\[
\lambda_r(q) = \lambda_{act}(a) \quad \text{whenever} \quad q = (a, \{ t_\sigma \mid \sigma \in \Sigma_{act} \}) \in Q_r.
\]

\( \delta_r : Q_r \times \Sigma \to 2^{Q_r} \) is the transition function of the TDES and is defined as follows.

Let \( q = (a, \{ t_\alpha \mid \alpha \in \Sigma_{act} \}) \). Then for \( \sigma \in \Sigma_{act} \), \( \delta(q, \sigma) \neq \emptyset \) if and only if

1. \( \sigma = \tau \), and \( t_\alpha > 0 \) for all prospective \( \alpha \)'s, or

2. \( \sigma \) is prospective, \( \delta_{act}(a, \sigma) \neq \emptyset \), and \( 0 \leq t_\sigma \leq u_\sigma - l_\sigma \), or

3. \( \sigma \) is remote, \( \delta_{act}(a, \sigma) \neq \emptyset \), and \( t_\sigma = 0 \).

Let \( q' = (a', \{ t'_\alpha \mid \alpha \in \Sigma_{act} \}) \in \delta(q, \sigma) \).

1. If \( \sigma = \tau \), then \( a' := a \), and

   for all \( \alpha \) prospective

   \[
t'_\alpha := \begin{cases} 
   u_\alpha & \text{if } \delta_{act}(a, \alpha) = \emptyset \\
   t_\alpha - 1 & \text{if } \delta_{act}(a, \alpha) \neq \emptyset \text{ and } t_\alpha > 0 
   \end{cases}
\]

   (Note that if \( t_\alpha = 0 \), then \( \delta(q, \tau) = \emptyset \))

   and for all \( \alpha \) remote

   \[
t'_\alpha := \begin{cases} 
   l_\alpha & \text{if } \delta_{act}(a, \alpha) = \emptyset \\
   t_\alpha - 1 & \text{if } \delta_{act}(a, \alpha) \neq \emptyset \text{ and } t_\alpha > 0 \\
   0 & \text{if } \delta_{act}(a, \alpha) \neq \emptyset \text{ and } t_\alpha = 0.
   \end{cases}
\]

2. If \( \sigma \in \Sigma_{act} \), then, by definition, \( a' \in \delta_{act}(a, \sigma) \), and

   for all \( \alpha \neq \sigma \) and \( \alpha \) prospective

   \[
t'_\alpha := \begin{cases} 
   u_\alpha & \text{if } \delta_{act}(a', \alpha) = \emptyset \\
   t_\alpha & \text{if } \delta_{act}(a', \alpha) \neq \emptyset,
   \end{cases}
\]

   \( t'_\sigma := u_\sigma \) if \( \sigma \) is prospective,
for all $\alpha \neq \sigma$ and $\alpha$ remote

$$t'_\alpha := \begin{cases} 
  l_\alpha & \text{if } \delta_{\text{act}}(a', \alpha) = \emptyset \\
  t_\alpha & \text{if } \delta_{\text{act}}(a', \alpha) \neq \emptyset,
\end{cases}$$

and $t'_\sigma = l_\sigma$ if $\sigma$ is remote.

In this framework, the activity transition model of a system consisting of several subsystems can be obtained from the synchronous product of the activity transition models of the subsystems [3] (Note that in our framework, the activity transition systems are nondeterministic). The time bounds of events in the resultant system are functions of the time bounds of the events in each subsystem. Suppose $G_{\text{act}}$ is obtained as the synchronous product of $G_{1,\text{act}}$ and $G_{2,\text{act}}$. Let $\Sigma_{1,\text{act}}$ and $\Sigma_{2,\text{act}}$ be the event sets of $G_{1,\text{act}}$ and $G_{2,\text{act}}$. If $\sigma \notin \Sigma_{1,\text{act}} \cap \Sigma_{2,\text{act}}$, then we usually take the time bounds of $\sigma$ to be identical to their values in the corresponding activity transition model. If $\sigma \in \Sigma_{1,\text{act}} \cap \Sigma_{2,\text{act}}$, then $l_\sigma$ and $u_\sigma$, the time bounds in $G_{\text{act}}$, will depend on $l_{1,\sigma}$, $u_{1,\sigma}$, $l_{2,\sigma}$ and $u_{2,\sigma}$ (the corresponding time bounds of $\sigma$ in $G_{1,\text{act}}$ and $G_{2,\text{act}}$) and on the interaction between the subsystems. The following rule has been suggested in [3] (although, in this thesis, we do not necessarily abide by it):

$$(l_\sigma, u_\sigma) = (\max(l_{1,\sigma}, l_{2,\sigma}), \min(u_{1,\sigma}, u_{2,\sigma})),$$

if $l_\sigma \leq u_\sigma$; otherwise the synchronous product is undefined. This rule embodies the idea that a shared event may be executed if and only if both subsystems agree on it.

The models $G_{\text{act}}$ and $G_r$ describe the behaviour of the system in both normal and faulty situations. Suppose there are $p$ failure modes $F_1, \ldots, F_p$. The event set $\Sigma_{\text{act}}$ includes failure events. As mentioned earlier, our discussion in this chapter will be limited to single-failure scenarios.

It is assumed that the activity set $A$ can be partitioned according to the condition of the system: $A = A_N \cup A_{F_1} \cup \cdots \cup A_{F_p}$. With $\mathcal{K} = \{N, F_1, \ldots, F_p\}$ as the condition set, the condition map of $G_{\text{act}}$, $\kappa_{\text{act}} : A \rightarrow \mathcal{K}$, is defined such that for every $a \in A$, $\kappa_{\text{act}}(a) = N$ if $a \in A_N$, and $\kappa(a) = F_i$ if $a \in A_{F_i}$ ($i \in \{1, \ldots, p\}$). The condition map
of $G_r$, $\kappa_r : Q_r \rightarrow K_r$ is defined in terms of $\kappa_{act}$; $\kappa_r(q) = \kappa_{act}(a)$ for all $q = (a, \{t_\sigma \mid \sigma \in \Sigma_{act}\})$.

**Remark 3.1** In the above ATG models, the lower and upper bounds of non–tick events are not affected by failure. In some practical cases, a failure may change these bounds; it may also delay or speed up the enabled events. One way of extending the ATG models to cover the above cases is (i) to let each non–tick event have different sets of time bounds for each condition at which the event can be enabled, and (ii) to make sure that the timers $t_\sigma$ stay within their permitted bounds. Specifically, consider a non–tick event $\alpha$. We let $\alpha$ have the time bounds $l_{\alpha,N}$ and $u_{\alpha,N}$ (resp. $l_{\alpha,F_i}$ and $u_{\alpha,F_i}$) if $\alpha$ can be enabled at the normal mode (resp. the failure mode $F_i$). In our framework, a change of condition is a result some failure event (or perhaps recovery event $^1$). Now consider the transition $q \xrightarrow{F_i} q'$ in the TDES which represents the occurrence of a failure $F_i$. If $\alpha$ is enabled at $q$ but not enabled at $q'$, then, as before, we let $t'_\alpha$, the value of the timer corresponding to $\alpha$ after the transition, be given by $t'_\alpha = u_\alpha$ if $\alpha$ is prospective and $t'_\alpha = l_\alpha$ if $\alpha$ is remote. If $\alpha$ is enabled at $q'$, then we let $t'_\alpha = f^i_\alpha(t_\alpha, l_\alpha, u_\alpha, l'_\alpha, u'_\alpha)$, where $t_\alpha$ is the value of $\alpha$’s timer before transition, $l_\alpha$ and $u_\alpha$ (resp. $l'_\alpha$ and $u'_\alpha$) are the time bounds of $\alpha$ in the condition $\kappa_r(q)$ (resp. $\kappa_r(q')$), and $f^i_\alpha$ is a function satisfying $0 \leq f^i_\alpha(t_\alpha, l_\alpha, u_\alpha, l'_\alpha, u'_\alpha) \leq u'_\alpha$ if $\alpha$ is prospective and $0 \leq f^i_\alpha(t_\alpha, l_\alpha, u_\alpha, l'_\alpha, u'_\alpha) \leq l'_\alpha$ if $\alpha$ is remote. $f^i_\alpha$ describes the effect of the failure mode $F_i$ on the timing of $\alpha$ and can be used to model delay or advance in the occurrence of $\alpha$. Synchronous product of ATG models can be defined as before. Here, of course, the dependency of time bounds on the condition (hence, state) should be taken into account. For example, suppose $G_{act}$ is obtained as the synchronous product of $G_{1,act}$ and $G_{2,act}$. If at a given activity of $G_{act}$ $a = (a_1, a_2)$ (with $a_1$ and $a_2$ being the corresponding activities in $G_{1,act}$ and $G_{2,act}$), $\alpha \in \Sigma_{1,act} \cap \Sigma_{2,act}$ is enabled, then the time bounds of $\alpha$, in general, depend on its time bounds in $\kappa_{1,act}(a_1)$ and $\kappa_{2,act}(a_2)$ ($\kappa_{1,act}$ and $\kappa_{2,act}$ are the condition maps of $G_{1,act}$ and $G_{2,act}$). In [4], another framework for having different sets of time bounds for each event is proposed which is simpler

---

1If failure recovery is also modelled, then the event set will include failure recovery events.
but not as general as the one proposed here.

From now on, we assume that the plant under control, $G_r$, is a TDES as defined at the beginning of this section, which may or may not have been obtained from an ATG model.

Alur et. al [1] have proposed another framework for capturing timing information in discrete–event systems. We have decided to use an approach based on TDES models for the following reasons:

- A useful range of problems in control engineering can be modelled using TDES;
- They are amenable to a useful subsystem composition as described earlier in this section (when the TDES are obtained from ATG models);
- This approach results in a considerably simpler theory for fault diagnosis because events are timed with respect to a single global clock while in [1], in general, several asynchronous timers are used for timing. Our framework for fault diagnosis may be reformulated in the setup of [1]; the resulting theory will probably be too complex to be useful practically.

Without loss of generality, we assume that no $\tau$ transition results in a change in the output. If the TDES is obtained from an activity transition system, then the above assumption will be true. In general, if the assumption fails, then we replace any transition $q_1 \xrightarrow{\tau} q_2$, with $\lambda_r(q_1) \neq \lambda_r(q_2)$, with the two transition

$q_1 \xrightarrow{\tau} q_1' \xrightarrow{\sigma_{12}} q_2$

with $\lambda_r(q_1') = \lambda_r(q_1)$. Here $q_1'$ and $\sigma_{12}$ are a new state and event that will be added to the state and event sets of the TDES. The resulting TDES will satisfy the above assumption.

Since $\tau$ transitions do not result in output change, it will be useful for diagnoser design to project out the clock ticks from the TDES. For example, consider a path $q_1 \xrightarrow{\tau} q_2 \xrightarrow{\tau} q_3 \xrightarrow{\sigma} q_4$ in the TDES. We would like to build a system in which the above path is replaced by a single transition $q_1 \xrightarrow{\sigma} q_4$ and store the transition time (2 ticks) in some set. This transition simply says that $q_4$ is reachable from $q_1$ through a sequence of clock ticks (in this case, 2) followed by a $\sigma$ transition. We refer to
the resulting system which describes the order and duration (in ticks) of occurrence of the non-tick events in the plant as the **timed finite-state Moore automaton** (corresponding to the TDES model). The timed FSMA corresponding to the TDES $G$ is defined to be $G = (X, \Sigma, \delta, \mathcal{T}, x_0, Y, \lambda)$, where $X$, $\Sigma$ and $Y$ are the state, event and output sets; $\delta$, $\mathcal{T}$ and $\lambda$ are the transition, transition-time and output functions, and $x_0$ is the initial state. By definition,

\[
X = \{q_0\} \cup \{x \in Q_\tau \mid \exists \sigma \in \Sigma_\tau - \{\tau\}, x' \in Q_\tau : x \in \delta_\tau(x', \sigma)\},
\]

\[
\Sigma = \Sigma_\tau - \{\tau\},
\]

\[
Y = Y_\text{act},
\]

\[
x_0 = q_0.
\]

$X$ is the set of states that the TDES can enter with a non-tick transition.

**Notation.** For every $x, x' \in X$ and $\sigma \in \Sigma$, we write $x \xrightarrow{\sigma} x'$ if and only if there exist $l \geq 2$, $q_1, \ldots, q_l$, with $q_1 = x$ and $q_l = x'$, such that $q_{i+1} \in \delta_\tau(q_i, \tau)$ for $1 \leq i \leq l - 2$ and $q_l \in \delta_\tau(q_{l-1}, \sigma)$.

The transition function $\delta : X \times \Sigma \rightarrow 2^X$ is defined according to:

\[
\forall x_1, x_2 \in X, \sigma \in \Sigma : x_2 \in \delta(x_1, \sigma) \iff x_1 \xrightarrow{\sigma} x_2.
\]

$\delta(x_1, \sigma)$ is the set of states that can be reached from $x_1$ through a sequence of $n$ consecutive $\tau$ transitions (for some $n \geq 0$), followed by a $\sigma$ transition.

We also define the transition–time function $\mathcal{T} : X \times \Sigma \times X \rightarrow 2^\mathbb{N}$ as follows:

\[
\forall x_1, x_2 \in X, \sigma \in \Sigma : \exists t \in \mathcal{T}(x_1, \sigma, x_2) \iff
\begin{cases}
\exists q_1, \ldots, q_t : q_1 \in \delta_\tau(x_1, \tau) & \\
((\forall i \in [1, t - 1]) q_{i+1} \in \delta_\tau(q_i, \tau)) & \land x_2 \in \delta_\tau(q_t, \sigma) \quad \text{if } t \geq 1,
\end{cases}
\]

\[
x_2 \in \delta_\tau(x_1, \sigma)
\]

if $t = 0$.

Obviously, if $x_2 \not\in \delta(x_1, \sigma)$, then $\mathcal{T}(x_1, \sigma, x_2) = \emptyset$. $\mathcal{T}(x_1, \sigma, x_2)$ is the set of times (in
Figure 3.1: TDES and the corresponding timed FSMA.

ticks) that a \( \sigma \)-transition from \( x_1 \) to \( x_2 \) may take in the timed FSMA.

The output map \( \lambda : X \rightarrow Y \) is given according to: \( \lambda(x) = \lambda_\tau(x) \), for all \( x \in X \subseteq Q_\tau \). We further define a condition map \( \kappa : X \rightarrow K \) with \( \kappa(x) := \kappa_\tau(x) \), for all \( x \in X \subseteq Q_\tau \), and extend the definition to all subsets \( z \subseteq X \) according to \( \kappa(z) := \bigcup \{ \kappa(x) \mid x \in z \} \).

For example, consider the TDES and the corresponding timed FSMA of Fig. 3.1. Here, \( Q_\tau = \{0, \ldots, 5\} \), \( q_0 = 0 \), \( \Sigma_\tau = \{\sigma_1, \sigma_2, \sigma_3, \tau\} \), \( Y = \{y_1, y_2\} \), \( X = \{0, 3, 4\} \), \( \mathcal{T}(0, \sigma_1, 3) = \{2\} \), \( \mathcal{T}(0, \sigma_2, 4) = \{3\} \), \( \mathcal{T}(0, \sigma_3, 4) = \{1\} \) and \( \mathcal{T}(3, \sigma_2, 4) = \{0\} \). The timed FSMA describes the order of occurrence and the timing of the non-tick events in the TDES.

Suppose \( x \in X \) and \( x \neq q_0 = x_0 \). Then there exist \( x' \in Q_\tau \) and \( \sigma \in \Sigma_\tau - \{\tau\} \) such that \( x \in \delta_\tau(x', \sigma) \). \( Q_\tau \) is reachable from \( q_0 \), by assumption. Therefore, there exists a path in \( G_\tau \) from \( q_0 \) to \( x' \) which can be continued by a \( \sigma \) transition to \( x \). After projecting out the \( \tau \) transitions of the path from \( q_0 \) to \( x \), we will obtain a path in \( G \) which starts at \( q_0 \) and ends with a \( \sigma \) transition to \( x \). This shows that in \( G \), \( X \) is reachable from \( x_0 = q_0 \).

Consider a path \( q_1 \xrightarrow{\tau} q_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_{t-1} \xrightarrow{\sigma} q_t \) in the TDES, with \( q_t \in X \) and \( t \geq 2 \). After projecting out the \( \tau \) transitions, we get a path \( q_1 \xrightarrow{\sigma} q_t \) in \( G \) with \( t - 2 \in \mathcal{T}(q_1, \sigma, q_t) \). Conversely, for any transition \( q \xrightarrow{\sigma} q' \) in \( G \), there exists a path in the TDES starting at \( q_1 \) and ending at \( q' \), consisting of some \( t \) back-to-back tick transitions \( (t \in \mathcal{T}(q, \sigma, q')) \) followed by a \( \sigma \) transition to \( q' \). For example, in Fig. 3.1, \( 0 \xrightarrow{\tau} 1 \xrightarrow{\tau} 2 \xrightarrow{\tau} 3 \xrightarrow{\sigma} 4 \) in the TDES is replaced by \( 0 \xrightarrow{\sigma} 4 \) in the timed FSMA, with
\( \tau(0, \sigma_2, 4) = \{3\} \).

Output-adjacent states in \( G_r \) and \( G \) are defined according to Def. 2.1. A very important point to remember is that if a state \( q \) in the TDES is output-adjacent to some \( q' \in Q_r \), then \( q \in X \) because, by assumption, the TDES output changes only as a result of non-tick transitions. The following lemma is easy to prove.

**Lemma 3.1** For \( q, q' \in X \), \( q \Rightarrow q' \) in \( G \) if and only if \( q \Rightarrow q' \) in \( G_r \). \( \Box \)

For example, in Fig. 3.1, the state 4 is output-adjacent to 0 and 3 in both the TDES and the timed FSMA.

**Remark 3.2** For any \( x, x' \in X \) and \( \sigma \in \Sigma \), \( \tau(x, \sigma, x') \) may be either a finite or an unbounded set. \( \tau(\cdot, \cdot, \cdot) \) becomes unbounded only when \( G_r \) contains cycles consisting of the tick event, \( \tau \). If \( G_r \) is obtained from an activity transition system, then \( \tau \) selfloops can be the only \( \tau \) cycles \( G_r \) may contain. This is because after every \( \tau \) transition, except in \( \tau \) selfloops, at least one of the timers \( t_\sigma (\sigma \in \Sigma) \) and hence, \( \sum_{\sigma \in \Sigma} t_\sigma \), decrease. As a result, in any sequence (other than a \( \tau \) selfloop) \( q_1 \xrightarrow{\tau} q_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_n \), the states \( q_i \) are all distinct.

Suppose \( G_r \) is obtained from an activity transition system. For \( x, x' \in X \) and \( \sigma \in \Sigma \), \( \tau(x, \sigma, x') \) will be unbounded if there exists a path \( x = q_1 \xrightarrow{\tau} q_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_{l-1} \xrightarrow{\tau} q_l \xrightarrow{\sigma} q_{l+1} = x' \) (\( l \geq 0 \)) with at least one of the states \( q_1, \ldots, q_l \) having a \( \tau \) selfloop. Therefore \( \tau(x, \sigma, x') \) will have one of the following forms:

(i) a finite set of integers, or

(ii) union of a finite set of integers and a set \([t^*, \infty)\), for some \( t^* \in \mathbb{N} \).

For any \( x \in X \), the sets \( \tau(x, \cdot, \cdot) \) can be calculated using a breadth-first search with a time complexity of \( O(|Q_r||T_{G_r}|) \), with \( T_{G_r} \) denoting the set of transitions of \( G_r \).

If, however, \( G_r \) is not obtained from an activity transition system and includes \( \tau \) cycles other than \( \tau \) selfloops, then the \( \tau(x, \sigma, x') \), in general, will have complicated forms. This issue will be further discussed in the following section. \( \Box \)
In this chapter, we use two running examples to illustrate our methodology.

**Example 3.1 - A Simple TDES**

Consider the TDES in Fig. 3.2. In this system, \( Q_\tau = \{0, \ldots, 9\} \), \( q_0 = 0 \), \( \Sigma_\tau = \{\sigma_1, \sigma_2, \sigma_3, \tau, F\} \), \( Y = \{o_1, o_2\} \), \( K = \{N, F\} \), \( Q_{\tau,N} = \{0, \ldots, 5\} \), and \( Q_{\tau,F} = \{6, \ldots, 9\} \). The system has a failure mode \( F \). In both normal and faulty modes, the output sequence generated by the system is \( \cdots o_1 o_2 o_1 o_2 \cdots \). The timing, however, is different in these modes. In the normal mode, \( o_1 \) is generated one tick after \( o_2 \) while in the faulty mode, \( o_1 \) occurs immediately after \( o_2 \). The corresponding timed FSMA is shown in Fig. 3.3. Here, \( X = \{0, 2, 4, 6, 9\} \). In the figure, the transition–time set \( T(\cdot, \cdot, \cdot) \) of each transition is written by the transition. For example, \( T(0, \sigma_1, 2) = \{1\} \). We can see that in this system, the state 4 is output–adjacent to 0 and 2 (in both the TDES and the timed FSMA). Similarly, 6 is output–adjacent to 4 and 9. \( \square \)
Our second example is more complex and is based on a neutralization process.

**Example 3.2 - Neutralization Process**

In a (simplified) neutralization process (Fig. 3.4), initially, all valves are closed and the tank is almost empty. The process starts by opening valve V1 and filling the reaction tank up to level $l_1$ with the chemical to be treated (here acid). Next, V1 is closed and the neutralizer (base) is added by opening valve V2. When the alkalinity (pH) of the solution reaches the normal range, V2 is closed and the tank contents are drained through valve V3. This completes a cycle of the process. Following this, another cycle will be started.

The events and the corresponding time bounds are given in Table 3.1. Two failure modes are considered here: valve V1 stuck open ($F_1$) and valve V1 stuck closed ($F_2$). For simplicity, it is assumed V1 gets stuck open (resp. closed) only when it is open (resp. closed). Sensor measurements are 'pH' and 'Level', with pH $\in \{a(\text{acid}), n(\text{neutral}), b(\text{base})\}$ and Level $\in \{L_0, L_1, L_2, L_3\}$.

The control sequence consists of 8 steps:

C1: Order V1 open;
    WAIT UNTIL pH=$a$ OR Level=$L_1$;
    IF pH=$a$ GO TO C2;
    ELSE GO TO C3

C2: WHEN Level=$L_1$ GO TO C4
<table>
<thead>
<tr>
<th>Event</th>
<th>Time bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>oi</td>
<td>valve i open</td>
</tr>
<tr>
<td>ci</td>
<td>valve i closed</td>
</tr>
<tr>
<td>$L_{ij}$</td>
<td>level change from $L_i$ to $L_j$</td>
</tr>
<tr>
<td>a2n</td>
<td>pH change from a (acid) to n (neutral)</td>
</tr>
<tr>
<td>n2a</td>
<td>pH change from n (neutral) to a (acid)</td>
</tr>
<tr>
<td>W2</td>
<td>wait for 2 clock ticks</td>
</tr>
<tr>
<td>$F_1$</td>
<td>valve V1 stuck open</td>
</tr>
<tr>
<td>$F_2$</td>
<td>valve V1 stuck closed</td>
</tr>
</tbody>
</table>

Table 3.1: Example 3.2. Events and their time bounds.

C3: WHEN pH=a GO TO C4

C4: WHEN Level=$L_2$ GO TO C5

C5: Order V1 closed;
Order V2 open;
WHEN pH=n GO TO C6

C6: Order V2 closed;
Order V3 open;
WHEN Level=$L_1$ GO TO C7

C7: WHEN Level=$L_0$ GO TO C8

C8: Order V3 closed;
WAIT for 2 clock ticks;
GO TO C1

Note that, by assumption, the controller does not have a way of knowing whether a valve is open or closed. It is also assumed that in addition to Level and pH, the current step in the control sequence is known to the diagnoser. Therefore $Y \subseteq \{C1, \cdots, C8\} \times \{L_0, L_1, L_2, L_3\} \times \{a, n, b\}$.

The ATG and the corresponding TDES describing the neutralization process are depicted in Figures 3.5 and 3.6 (assuming $l = 2$, $u = 3$, $l' = 1$ and $u' = 2$ for the
time bounds of the $L_{ij}$ as given in Table 3.1). The output and the condition maps of the TDES are given in Table 3.2. The names of the TDES states have the form $q$ or $q.n$, where $q$ refers to the activity $q$ in the ATG (Fig. 3.5). The ATG consists of three subautomata each describing the behaviour of the system in the $N$ or $F_1$ or $F_2$ condition. Note that, to avoid cluttering the graph, we have depicted the transitions from the normal mode $N$ to the faulty conditions $F_1$ and $F_2$ (i.e., failure events) only on the subautomata corresponding to the faulty conditions. Initially, the system is at $q_0 = 1$ where the controller enables the event $o_1$ (open $V_1$). The lower and upper bounds of $o_1$ are both 1 which means that following its enablement, $o_1$ cannot occur before the first clock tick but if it remains enabled, it will certainly happen before the second clock tick. The event $o_1$ occurs after the first tick but before the second. Following this, the level reaches the $L_1$ range and the solution becomes acid within the time bounds specified in Table 3.1. Then level reaches $L_2$ in $l(= 2)$ to $u(= 3)$ clock ticks and, so on. Note that at state 15 when the controller orders $V_3$ closed, it waits two ticks of the clock before starting a new cycle to make sure $V_3$ is closed at the start of the new cycle. While $V_1$ is open, it may fail stuck-open which ultimately results in the occurrence of $L_{23}$. Also it may fail stuck-closed while it is closed. In this case, at the beginning of the new cycle when the controller orders $V_1$ open, neither $L_{01}$ nor $n2a$ will happen.

The timed FSM of the neutralization process is shown in Figures 3.7 and 3.8. According to the timed FSM, for example, the transition from state 5 to 6 takes 2 to 3 ticks. This is in accordance with the TDES (Fig. 3.6).

### 3.2 Diagnoser

For diagnoser design, it is possible to start with the TDES model of the system and, treating the clock tick as an extra output signal (i.e., extending the output map to include the ticks), design a diagnoser using the methodology presented in the previous chapter. This diagnoser, which we will refer to as the **standard diagnoser**, provides updates of the system's condition after the generation of every new output symbol in
Figure 3.5: Example 3.2. Activity transition graph.
Figure 3.6: Example 3.2. TDES model.
<table>
<thead>
<tr>
<th>State</th>
<th>Output, Condition</th>
<th>State</th>
<th>Output, Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(C1,L₀,n), N</td>
<td>2.1'</td>
<td>(C1,L₀,n), F₁</td>
</tr>
<tr>
<td>1.1</td>
<td>(C₁,L₀,n), N</td>
<td>2.2'</td>
<td>(C₁,L₀,n), F₁</td>
</tr>
<tr>
<td>2</td>
<td>(C₁,L₀,n), N</td>
<td>3'</td>
<td>(C₂,L₀,a), F₁</td>
</tr>
<tr>
<td>2.1</td>
<td>(C₁,L₀,n), N</td>
<td>4'</td>
<td>(C₃,L₁,n), F₁</td>
</tr>
<tr>
<td>3</td>
<td>(C₂,L₀,a), N</td>
<td>5'</td>
<td>(C₄,L₁,a), F₁</td>
</tr>
<tr>
<td>4</td>
<td>(C₃,L₁,n), N</td>
<td>5.1'</td>
<td>(C₄,L₁,a), F₁</td>
</tr>
<tr>
<td>5</td>
<td>(C₄,L₁,a), N</td>
<td>5.2'</td>
<td>(C₄,L₁,a), F₁</td>
</tr>
<tr>
<td>5.1</td>
<td>(C₄,L₁,a), N</td>
<td>5.3'</td>
<td>(C₄,L₁,a), F₁</td>
</tr>
<tr>
<td>5.2</td>
<td>(C₄,L₁,a), N</td>
<td>6'</td>
<td>(C₅,L₂,a), F₁</td>
</tr>
<tr>
<td>5.3</td>
<td>(C₄,L₁,a), N</td>
<td>6.1'</td>
<td>(C₅,L₂,a), F₁</td>
</tr>
<tr>
<td>6</td>
<td>(C₅,L₂,a), N</td>
<td>8'</td>
<td>(C₅,L₂,a), F₁</td>
</tr>
<tr>
<td>6.1</td>
<td>(C₅,L₂,a), N</td>
<td>8.1'</td>
<td>(C₅,L₂,a), F₁</td>
</tr>
<tr>
<td>7</td>
<td>(C₅,L₂,a), N</td>
<td>8.2'</td>
<td>(C₅,L₂,a), F₁</td>
</tr>
<tr>
<td>8</td>
<td>(C₅,L₂,a), N</td>
<td>17''</td>
<td>(C₅,L₃,a), F₁</td>
</tr>
<tr>
<td>9</td>
<td>(C₅,L₂,a), N</td>
<td>7''</td>
<td>(C₅,L₂,a), F₂</td>
</tr>
<tr>
<td>9.1</td>
<td>(C₅,L₂,a), N</td>
<td>9''</td>
<td>(C₅,L₂,a), F₂</td>
</tr>
<tr>
<td>9.2</td>
<td>(C₅,L₂,a), N</td>
<td>9.1''</td>
<td>(C₅,L₂,a), F₂</td>
</tr>
<tr>
<td>9.3</td>
<td>(C₅,L₂,a), N</td>
<td>9.2''</td>
<td>(C₅,L₂,a), F₂</td>
</tr>
<tr>
<td>10</td>
<td>(C₆,L₂,n), N</td>
<td>9.3''</td>
<td>(C₅,L₂,a), F₂</td>
</tr>
<tr>
<td>10.1</td>
<td>(C₆,L₂,n), N</td>
<td>10''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>11</td>
<td>(C₆,L₂,n), N</td>
<td>10.1''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>12</td>
<td>(C₆,L₂,n), N</td>
<td>11''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>13</td>
<td>(C₆,L₂,n), N</td>
<td>12''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>13.1</td>
<td>(C₆,L₂,n), N</td>
<td>13''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>13.2</td>
<td>(C₆,L₂,n), N</td>
<td>13.1''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>13.3</td>
<td>(C₆,L₂,n), N</td>
<td>13.2''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>14</td>
<td>(C₇,L₁,n), N</td>
<td>13.3''</td>
<td>(C₆,L₂,n), F₂</td>
</tr>
<tr>
<td>14.1</td>
<td>(C₇,L₁,n), N</td>
<td>14''</td>
<td>(C₇,L₁,n), F₂</td>
</tr>
<tr>
<td>14.2</td>
<td>(C₇,L₁,n), N</td>
<td>14.1''</td>
<td>(C₇,L₁,n), F₂</td>
</tr>
<tr>
<td>14.3</td>
<td>(C₇,L₁,n), N</td>
<td>14.2''</td>
<td>(C₇,L₁,n), F₂</td>
</tr>
<tr>
<td>15</td>
<td>(C₈,L₀,n), N</td>
<td>14.3''</td>
<td>(C₇,L₁,n), F₂</td>
</tr>
<tr>
<td>15.1</td>
<td>(C₈,L₀,n), N</td>
<td>15''</td>
<td>(C₈,L₀,n), F₂</td>
</tr>
<tr>
<td>16</td>
<td>(C₈,L₀,n), N</td>
<td>15.1''</td>
<td>(C₈,L₀,n), F₂</td>
</tr>
<tr>
<td>16.1</td>
<td>(C₈,L₀,n), N</td>
<td>16''</td>
<td>(C₈,L₀,n), F₂</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.1''</td>
<td>(C₈,L₀,n), F₂</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1''</td>
<td>(C₁,L₀,n), F₁</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1''</td>
<td>(C₁,L₀,n), F₁</td>
</tr>
</tbody>
</table>

Table 3.2: Example 3.2. Output and condition maps of TDES.
Figure 3.7: Example 3.2. Timed FSMA (Part 1).
Figure 3.8: Example 3.2. Timed FSMA (Part 2).
the output set $Y$ and every clock tick. In [4], a similar method has been developed for the extension of the event–based framework of [29]. This is a straightforward approach. However, the number of states of the corresponding RTS and diagnoser will be very large due to the incorporation of timing information.

In this chapter, we propose an alternative approach in which the process of updating the estimate of the system’s condition is performed only when a new output symbol $y \in Y$ is generated. The update process is based on the generated output symbols and the number of clock ticks occurring between them; no updating at clock ticks is required in this method. As will be shown later, for fault detection and diagnosis between two consecutive output symbols, we can rely on what we refer to as predictions of the system’s condition. This results in significant reduction in on-line computing requirements and, in many cases, in the size of the diagnoser, albeit at the expense of extra off-line design calculations. Because of the use of predictions in diagnosis, we refer to this diagnoser as the predicting diagnoser. In the following, we first discuss diagnoser construction. Then we will explain how the diagnoser can be used for fault diagnosis.

- **Diagnoser Construction**

Consider the plant under control which in the previous section, was modelled as a timed FSMA. This timed FSMA generates an output sequence $y_1, y_2, \cdots, y_k$. Let $t_k$ denote the number of clock ticks that occurred between the observations $y_{k-1}$ and $y_k$. The diagnoser proposed in this chapter generates a state estimate $z_k \in 2^X - \{\emptyset\}$ for the timed FSMA based on the output sequence $y_1, y_2, \cdots, y_k$ and the timing sequence $t_2, \cdots, t_k$. $\kappa(z_k)$ will be the estimate of the system’s condition at the time that $y_k$ was generated (Fig 3.9). After $t_{k+1}$ ticks and upon observing $y_{k+1}$, $z_k$ will be updated to $z_{k+1}$.

The functions $\hat{T}_r : Q_r \times Q_r \to 2^N$ and $\tilde{T} : X \times X \to 2^N$, defined in the following, give the possible transition times between output–adjacent states in the TDES and
the timed FSMA, respectively.

\[
\tilde{T}_\tau(q, q') := \begin{cases} 
\{t \mid t \text{ is the time (in ticks) it can take the TDES} \\
\text{to go from } q \text{ to } q' \text{ using a path along} \\
\text{which the output is } \lambda_\tau(q) \text{ (except at } q') \} & \text{if } q \Rightarrow q', \\
\emptyset & \text{otherwise.}
\end{cases}
\]

\[
\tilde{T}(x, x') := \begin{cases} 
\{t \mid t \text{ is the time (in ticks) it can take the timed} \\
\text{FSMA to go from } x \text{ to } x' \text{ using a path along} \\
\text{which the output is } \lambda(x) \text{ (except at } x') \} & \text{if } x \Rightarrow x', \\
\emptyset & \text{otherwise.}
\end{cases}
\]

These functions are used in the diagnoser state update law. For example, in the neutralization process, \(1 \Rightarrow 3\), \(\tilde{T}_\tau(1, 3) = \tilde{T}(1, 3) = \{2\}\); \(9 \Rightarrow 10''\), \(\tilde{T}_\tau(9, 10'') = \tilde{T}(9, 10'') = \{2, 3\}\); \(9.2 \Rightarrow 10'' \text{ (in } G_\tau)\), and \(\tilde{T}_\tau(9.2, 10'') = \{0, 1\}\). Note that for \(x, x' \in X, x \Rightarrow x' \text{ in } G\) if and only if \(x \Rightarrow x' \text{ in } G_\tau\) (Lemma 3.1), and \(\tilde{T}(x, x') = \tilde{T}_\tau(x, x')\).

We define the **predicting diagnoser** to be a finite-state machine \(D = (Z, \hat{Y}, \zeta, z_0, \hat{K}, \kappa)\), where \(Z \subseteq (2^X - \{\emptyset\}) \times \{0, 1, 2\}, \hat{Y} \subseteq Y \times \mathbb{N}, \hat{K} \subseteq 2^K - \{\emptyset\}\) are the state, event and output sets of \(D\). Note that the event set can be countably infinite. \(z_0\) is the initial state, \(\zeta : Z \times \hat{Y} \rightarrow Z\) is the transition (partial) function, and \(\kappa : Z \rightarrow \hat{K}\) the output map.

The state of \(D\) has the general form \(z = (z, e)\), where \(z \in 2^X - \{\emptyset\}\) and \(e \in \{0, 1, 2\}\). We will explain the role of \(e\), the second component of \(z\), later, after discussing the transition function of the diagnoser. For the initial state of the diagnoser \(e = 0\); i.e., \(z_0 = (z_0, 0)\), for some \(z_0 \in 2^X - \{\emptyset\}\). \(z_0\) contains the information available about the
state of the system $G$ before the diagnoser is initialized. Here we assume that the state of the TDES before the initiation of diagnosis is

$$z_{\tau,0} = z_0 \cup \{q \in Q_\tau \mid q \text{ is reachable from a state } x \in z_0 \text{ through a sequence of } \tau \text{ transitions only}\}.$$

This means that at the time when the diagnoser is initialized, no information about the time of generation of the latest event (and output) is available.

Usually $z_0 = X$ (therefore, $z_{\tau,0} = Q_\tau$) because the diagnoser may be initialized at any time while the system is in operation and, in this situation, the state of the system is usually unknown. If the system is only known to be normal at the time that the diagnoser is initialized, then $z_0 = X_N$.

After the diagnoser is started and the first output $y_1$ is read, $z_0$ is updated to $z_1 = z_0 \cap \lambda^{-1}\{\{y_1\}\}$. Since $\tau$ transitions do not result in output change, $z_{\tau,1}$, the state of the TDES at the time $y_1$ is read, is

$$z_{\tau,1} = z_{\tau,0} \cap \lambda^{-1}\{\{y_1\}\} = z_1 \cup \{q \in Q_\tau \mid q \text{ is reachable from a state } x \in z_1 \text{ through a sequence of } \tau \text{ transitions only}\}.$$

The diagnoser state transition $z_{k+1} = (z_{k+1}, e_{k+1}) = \zeta(z_k, y_{k+1}, t_{k+1})$ is defined according to

$$\begin{cases} z_1 = z_0 \cap \lambda^{-1}\{\{y_1\}\}, \\
e_1 = 1, \\
z_2 = \lambda^{-1}\{\{y_2\}\} \cap \{q \in Q_\tau \mid \exists q' \in z_{\tau,1} : q' \Rightarrow q \land t_2 \in \bar{T}_\tau(q', q)\}, \\
e_2 = 2, \\
z_{k+1} = \lambda^{-1}\{\{y_{k+1}\}\} \cap \{x \mid \exists x' \in z_k : x' \Rightarrow x \land t_{k+1} \in \bar{T}(x', x)\}, \\
e_{k+1} = 2, \quad \text{for } k \geq 2. \end{cases}$$
Note that \( z_1 \) only depends on \( z_0 \) and \( y_1 \); \( t_1 \) is actually undefined. The diagnoser output map \( \kappa : Z \rightarrow \hat{K} \) gives the estimate of the system’s condition:

\[
\kappa(z_k) = \kappa((z_k, e_k)) := \kappa(z_k).
\]

As mentioned earlier, \( z_0 \) is the estimate of the system’s state before the diagnoser is initialized. Upon starting the diagnoser and observing the output symbol \( y_1 \), the state estimate is updated to \( z_1 \). At this time, the state of the TDES is in \( z_{r,1} \). If after \( t_2 \) ticks \( y_2 \) is generated, then \( z_1 \) will be updated to \( z_2 \). In order to obtain \( z_2 \), first we find the set of states \( q \) that are output-adjacent to some \( q' \in z_{r,1} \), and satisfy \( t_2 \in \bar{T}_r(q', q) \). Note that this set is a subset of \( X \) (because each of the states of the set is output-adjacent to some state). Then \( z_2 \) will be those \( q \)'s in the above set that have \( y_2 \) as output. Next, if \( y_3 \) is generated after \( t_3 \) ticks, then \( z_3 \), the current state estimate, will be the set of states \( x \in \lambda^{-1}(\{y_3\}) \) that are output-adjacent to some \( x' \in z_2 \) with \( t_3 \in \bar{T}(x', x) \). \( z_3 \subseteq X \) is the estimate for the state of the timed FSMA (and the TDES) when \( y_3 \) is generated. The diagnoser keeps updating the state estimate in this way after every occurrence of new output symbols from the set \( Y \). For \( k \geq 2 \), \( z_k \) is the estimate for the state of the FSMA (and the TDES) at the time \( y_k \) is generated.

At step \( k \), \( z_k \) is the latest state estimate. Note that the update law for \( z_k \) is different for \( k = 0, k = 1 \) and \( k \geq 2 \). To take this into account, we have added a component \( e_k \) to the state of the diagnoser with \( e_0 = 0, e_1 = 1 \), and \( e_k = 2 \) for \( k \geq 2 \). This change in update law is the result of the fact that we do not require the system and the diagnoser to be initialized at the same time and that we do not assume any specific knowledge about the state and the condition of the system before the initiation of diagnosis.

As with fault diagnosis in finite-state automata, it is computationally economical to construct a transition system to summarize and store the information about the output-adjacent states of the timed FSMA. The timed reachability transition system corresponding to the timed FSMA \( G \) is defined to be \( \bar{G} = (X, \bar{T}, Y, \lambda) \), with
Timed reachability transition system.

State | Output-adjacent states \{time\} \\
--- | --- \\
0   | 4 \{2\} \\
2   | 4 \{1\} \\
4   | 0 \{1\}, 6 \{0\} \\
6   | 9 \{2\} \\
9   | 6 \{0\} \\

Table 3.3: Example 3.1. Timed reachability transition system.

$X$ and $Y$ as the state and the output sets and $\lambda : X \to Y$ the output map. The function $\mathcal{T} : X \times X \to 2^N$ was previously defined in this section; we shall refer to it as the transition–time function of the timed RTS. Note that if a state $x'$ is not output–adjacent to another state $x$, then $\mathcal{T}(x, x') = \emptyset$.

**Example 3.1 - A Simple TDES (Cont'd)**

The timed RTS and the diagnoser for this example are given in Table 3.3 and Fig. 3.10. Here we have assumed $z_0 = X_N = \{0, 2, 4\}$; i.e., the system is known to be normal when the diagnosis is initiated. This means that $z_{r,0} = \{0, 1, 2, 3, 4, 5\}$. If the first output symbol read, $y_1$, is $o1$, then $z_1 = \{0, 2\}$ and $z_{r,1} = \{0, 1, 2, 3\}$. In this case, the second output symbol $y_2$ has to be $o2$ and $z_2 = \{4\}$. At the time of the initiation of diagnosis, the TDES is in one of the states of $z_{r,1}$. The reader can verify that $\mathcal{T}_r(0, 4) = \{2\}$, $\mathcal{T}_r(1, 4) = \mathcal{T}_r(2, 4) = \{1\}$, and $\mathcal{T}_r(3, 4) = \{0\}$. Therefore, $o2$ can take 0 to 2 clock ticks to occur following the initialization of the diagnoser. The time interval $[0, 2]$ is written on the transition from $z = (\{0, 2\}, 1)$ to $z = (\{4\}, 2)$ to indicate this. Similarly, if $y_1 = o2$, then $z_1 = \{4\}$, $z_{r,1} = \{4, 5\}$ and we must have $y_2 = o1$. If $o1$ is generated after $o2$, before the next tick, then $z_2 = \{0, 6\}$. But if $o1$ is generated after one clock tick, then $z_2 = \{0\}$. The rest of the diagnoser can be constructed using the timed RTS and the diagnoser state transition law (for $k \geq 2$). Note that whenever an $o1$ is generated immediately after an $o2$, the diagnoser enters an $F$–certain state unless the $o2$ symbol is the first output symbol read.  

**Example 3.2 - Neutralization Process (Cont’d)**

The timed RTS for the neutralization process is given in Table 3.4, assuming $l = 2$,  

---

68
Figure 3.10: Example 3.1. Diagnoser.
\( u = 3, l' = 1 \) and \( u' = 2 \). In this table, the transition-time sets \( \hat{T}(.,.) \) are only provided for output-adjacent states. Also, '/-' is used to separate states with different outputs.

The diagnoser for the neutralization process is shown in Fig. 3.11 (assuming \( z_0 = X \)). Note that there are 4 parameters in the diagnoser: \( l, u, l' \) and \( u' \). The way the diagnoser operates is described in the following. If, for example, the first output symbol is \((C5,L2,a)\), then \( z_1 = \{6,7,8,9,6',6.1',8',7'',9'',9.1'', \cdots, 9.u''\} \).

Observe that \( z_{7,1} = z_1 \cup \{6.1,9.1,9.2,9.3,8.1',8.2'\} \) (Note that, for example, \( 6.1' \in z_1 \) but \( 6.1 \notin z_1 \) since \( 6.1' \) can be entered with an \( F_1 \) transition, and therefore \( 6.1' \in X = z_0 \), while \( 6.1 \notin X = z_0 \) because it can be entered by \( \tau \) transitions only). If the next output symbol \( y_2 \) is \((C5,L3,a)\), then \( z_2 = \{17'\} \) and \( \kappa(z_2) = \{F_1\} \); hence, the diagnoser will reach an \( F_1 \)-certain state. According to the timed RTS, \((C5,L3,a)\) can be generated within \( l' \) to \( u' + 1 \) ticks of the clock from the states in \( z_1 \). Therefore \( y_2 \) can take any time between 0 to \( u' + 1 \) ticks to occur following the initialization of the diagnoser from the states in \( z_{7,1} \); i.e., \( 0 \leq t_2 \leq u' + 1 \). The time interval \([0,u'+1]\) is written on the transition from \( \{6,7,8,9,6',6.1',8',7'',9'',9.1'', \cdots, 9.u''\} \) to \( \{17'\} \) to indicate this. Similarly, if instead of \((C5,L3,a)\), \( y_2 = (C6,L_2,n) \), then \( z_2 = \{10,10''\} \). This transition should occur in \( 0 \) to \( u+1 \) ticks. After this, \( y_3 = (C7,L_1,n) \) will be generated in \( l + 1 \) to \( u + 1 \) ticks \( (l + 1 \leq t_3 \leq u + 1) \). The rest of Fig. 3.11 can be similarly interpreted. The transition to \( z = \{1'',1.1''\} \) depicted by dashed lines on the left of Fig. 3.11 will be explained later in this section.

\[ \square \]

**Remark 3.3**

- If \( x' \) is output-adjacent to \( x \), then \( \hat{T}(x,x') \neq \emptyset \). In this case, \( \hat{T}(x,x') \) may be either a finite set or an unbounded set (when there are cycles in \( G_r \) and \( G \)). When the \( \hat{T}(.,.) \) are large and complicated sets, and therefore difficult to calculate, a design based on the standard diagnoser may be a more efficient solution to the fault diagnosis problem.

- Instead of representing the \( T \) and \( \hat{T} \) with their individual members, one may use more efficient ways for storing and manipulating sets such as representing...
<table>
<thead>
<tr>
<th>State</th>
<th>Output-adjacent states {time}</th>
<th>State</th>
<th>Output-adjacent states {time}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.3' {2} / 4.4' {2}</td>
<td>1''</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3.3' {1} / 4.4' {1}</td>
<td>1.1''</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>5.5' {0}</td>
<td>7''</td>
<td>10'' {2,3}</td>
</tr>
<tr>
<td>4</td>
<td>5.5' {0}</td>
<td>9''</td>
<td>10'' {2,3}</td>
</tr>
<tr>
<td>5</td>
<td>6.6' {2,3}</td>
<td>9.1''</td>
<td>10'' {1,2}</td>
</tr>
<tr>
<td>6</td>
<td>10,10'' {3,4} / 17'' {2,3}</td>
<td>9.2''</td>
<td>10'' {0,1}</td>
</tr>
<tr>
<td>7</td>
<td>10,10'' {2,3}</td>
<td>9.3''</td>
<td>10'' {0}</td>
</tr>
<tr>
<td>8</td>
<td>10 {2,3} / 17'' {1,2}</td>
<td>10''</td>
<td>14'' {3,4}</td>
</tr>
<tr>
<td>9</td>
<td>10,10'' {2,3}</td>
<td>10.1''</td>
<td>14'' {2,3}</td>
</tr>
<tr>
<td>10</td>
<td>14,14'' {3,4}</td>
<td>11''</td>
<td>14'' {2,3}</td>
</tr>
<tr>
<td>11</td>
<td>14,14'' {2,3}</td>
<td>12''</td>
<td>14'' {2,3}</td>
</tr>
<tr>
<td>12</td>
<td>14,14'' {2,3}</td>
<td>13''</td>
<td>14'' {2,3}</td>
</tr>
<tr>
<td>13</td>
<td>14,14'' {2,3}</td>
<td>13.1''</td>
<td>14'' {1,2}</td>
</tr>
<tr>
<td>14</td>
<td>15,15'' {2,3}</td>
<td>13.2''</td>
<td>14'' {0,1}</td>
</tr>
<tr>
<td>15</td>
<td>1.1'' {2}</td>
<td>13.3''</td>
<td>14'' {0}</td>
</tr>
<tr>
<td></td>
<td>1.1'' {1}</td>
<td>14''</td>
<td>15'' {2,3}</td>
</tr>
<tr>
<td>2.1'</td>
<td>3' {1} / 4' {1}</td>
<td>14.1''</td>
<td>15'' {1,2}</td>
</tr>
<tr>
<td>2.2'</td>
<td>3' {0} / 4' {0}</td>
<td>14.2''</td>
<td>15'' {0,1}</td>
</tr>
<tr>
<td>3'</td>
<td>5' {0}</td>
<td>14.3''</td>
<td>15'' {0}</td>
</tr>
<tr>
<td>4'</td>
<td>5' {0}</td>
<td>15''</td>
<td>1'' {2}</td>
</tr>
<tr>
<td>5'</td>
<td>6' {2,3}</td>
<td>15.1''</td>
<td>1'' {1}</td>
</tr>
<tr>
<td>5.1'</td>
<td>6' {1,2}</td>
<td>16''</td>
<td>1'' {1}</td>
</tr>
<tr>
<td>5.2'</td>
<td>6' {0,1}</td>
<td>16.1''</td>
<td>1'' {0}</td>
</tr>
<tr>
<td>5.3'</td>
<td>6' {0}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6'</td>
<td>17' {2,3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.1'</td>
<td>17' {1,2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8'</td>
<td>17' {1,2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4: Example 3.2. Timed reachability transition system.
Figure 3.11: Example 3.2. Diagnoser.
sets as unions of intervals and specifying intervals only with their lower and upper limits (rather than with their members).

- In order to reduce complexity, one may approximate sets with intervals (e.g., approximating \{1,2,3,4,6,7,8,9\} with \([1,9]\)). This results in diagnosers that will be less accurate, i.e., it may take the diagnoser a longer time to detect and isolate a failure, or, in the worst case, the diagnoser may not be able to detect or isolate a diagnosable failure at all. Assuming that the failures remain diagnosable, the computation of the lower and upper limits of the approximating intervals (corresponding to minimum and maximum times of the generation of events and outputs), reduces to the calculation of optimal paths between states in subgraphs of the TDES and the timed FSMA. For these calculations, optimization techniques such as dynamic programming can be used which are more efficient than brute-force calculation.

- It should be noted that complicated or unpredictable temporal behaviour (e.g., cycles) is generally considered undesirable in real-time control. Therefore, we can expect the methodology proposed here to be applicable to a very useful range of control problems, and to provide more efficient solutions compared with the standard diagnoser.

Using the Diagnoser

The diagnoser provides estimates of the system's condition at the instants when new output symbols are generated. For timely and accurate diagnosis, we may also need condition estimates between output symbols after every clock tick. In the following, we will discuss how for diagnosing permanent failures, these estimates can be replaced by predictions of the system's condition.

Let us assume that at some point an output symbol \(y\) is generated. Suppose \(z\) and \(\kappa(z)\) are the estimates of the system's state and condition following the generation of \(y\). Let \(\text{Rch}(z)\) denote the set of states of the timed FSMA that have output \(y\) and are
reachable from a state in \( z \) using a path along which the output is \( y \), namely,

\[
\text{Rch}(z) := \{ x \in X \mid \exists l \geq 1, \exists x_1, \ldots, x_l, \exists \sigma_1, \ldots, \sigma_l : x_1 \in z \& (\lambda(x_i) = \lambda(x) (1 \leq i \leq l)) \& (x_{i+1} \in \delta(x_i, \sigma_i) (1 \leq i \leq l-1)) \& x \in \delta(x_l, \sigma_l) \}.
\]

Also define

\[
\text{Lim}_1(z) := \{ x \in X \mid x \in \text{Rch}(z) \& (\delta(x, \sigma) = \emptyset, \text{ for any } \sigma \in \Sigma) \},
\]

\[
\text{Lim}_2(z) := \{ x \in X \mid x \in \text{Rch}(z) \& (\exists x' \in X, \sigma \in \Sigma : \sup T(x, \sigma, x') = \infty) \},
\]

\[
\text{Lim}_3(z) := \{ x \in X \mid x \text{ belongs to a cycle in } \text{Rch}(z) \},
\]

\[
\text{Lim}(z) := \text{Lim}_1(z) \cup \text{Lim}_2(z) \cup \text{Lim}_3(z).
\]

\( \text{Lim}_1(z) \) is the set of deadlock states in \( \text{Rch}(z) \) while \( \text{Lim}_2(z) \) is the set of states of \( \text{Rch}(z) \) from which at least one transition may take an arbitrarily long time to occur.

For example, in the neutralization process, \( \text{Rch} \{1,1^n\} = \{1,2,2.1',2.2',1'',1.1''\} \), \( \text{Lim} \{1,1^n\} = \text{Lim}_1 \{1,1^n\} = \{1'',1.1''\} \). In Example 3.1, however, \( \text{Lim}(z) = \emptyset \), for all state estimates \( z \) because the TDES never stops generating new output symbols.

**Lemma 3.2** Let \( x_1 \xrightarrow{t_1} x_2 \xrightarrow{t_2} \cdots \) be a path in the timed FSMA that starts in \( z \) (i.e., \( x_1 \in z \)) and stays in \( \text{Rch}(z) \) for all future time, with \( t_{i,i+1} \in \mathbb{N} \) denoting the time (in clock ticks) taken by the transition \( x_i \rightarrow x_{i+1} \). There exists \( N \geq 0 \) (in general, depending on the path) such that for \( i > N \), \( x_i \in \text{Lim}(z) \); therefore, the state enters \( \text{Lim}(z) \) in the finite time of at most \( \sum_{i=1}^{N} t_{i,i+1} \) clock ticks.

**Proof.** If \( \{x_i\} \) is a finite sequence consisting of, say, \( n_f \) transitions, then the system reaches the state \( x_{n_f+1} \) and stays there indefinitely. Therefore, \( x_{n_f+1} \) must be either in \( \text{Lim}_1(z) \) or in \( \text{Lim}_2(z) \). Thus, the lemma is true with \( N = n_f \).

If, on the other hand, \( \{x_i\} \) is an infinite sequence, then for \( i > n_{\text{max}} \), \( x_i \) must belong to \( \text{Lim}_3(z) \), where \( n_{\text{max}} = \max \{ n \mid x_n \in \{x_i\} \& x_n \in \text{Rch}(z) \& x_n \text{ does not belong to any cycle in } \text{Rch}(z) \} \). Note that since the states that
do not belong to the cycles in $Rch(z)$ can appear at most once in $\{x_i\}$ and $X$ is finite, $n_{\text{max}}$ is a finite number. Therefore, the lemma holds with $N = n_{\text{max}}$. \hfill \qed

Therefore, $\text{Lim}(z)$ is the set of states and cycles in $Rch(z)$ in which the system can be trapped for an arbitrarily long time without generating new output symbols.

Now consider Fig. 3.12 which depicts part of a diagnoser. According to the figure, following $y$, either a new output symbol $y'_i$ ($i \in \{1, 2, 3\}$) is generated after some time, in which case $x \in z'_i$ at the time $y'_i$ is observed, or no new output is generated and, hence, $x$ must eventually enter $\text{Lim}(z)$ after a finite time and stay there for all future time (by Lemma 3.2). For this reason, we refer to $\text{Lim}(z) \cup (\bigcup_{i=1}^{3} z'_i)$ as the prediction of the system's state at the time $y$ was generated. If four ticks occur without the generation of any new output symbol, we can conclude that the next output symbol cannot be $y'_i$. Therefore the transition to $z'_i$ can be ruled out and the prediction can be updated to $\text{Lim}(z) \cup z'_i \cup z'_j$.

We denote the set of transition times in the diagnoser by $\mathcal{T}(\ldots)$.

**Definition 3.1** Suppose $z, z' \in \mathbb{Z} - \{z_0\}$ and $y' = \lambda(x')$ for all $x' \in z'$. The transition-time function of the diagnoser, $\mathcal{T} : \mathbb{Z} - \{z_0\} \times \mathbb{Z} - \{z_0\} \rightarrow \mathbb{N}$, is defined according to

\[
\mathcal{T}(z, z') := \{ t | z' = \zeta(z, y', t) \}.
\]
Note that \( y' \) is a function of \( z' \); therefore \( \bar{T} \) depends explicitly only on \( z \) and \( z' \), and not on \( y' \). In Fig. 3.12, for example, \( \bar{T}(z, z') = \{2, 3\} \). In general, \( \bar{T}(z, z') \) might be empty for some \( z \) and \( z' \).

Formally, prediction of the state of the timed FSMA \( G \) is defined as follows.

**Definition 3.2** Suppose \( z (\neq z_0) \) is a diagnoser state and \( z \) the corresponding state estimate. The prediction of the system's state, \( t \) clock ticks after the diagnoser enters \( z \), assuming the generation of no new output symbols after \( y \), is defined according to

\[
\text{Pred}(z, t) := \text{Lim}(z) \cup X_{OA}(z, t),
\]

where

\[
X_{OA}(z, t) := \bigcup \{ z' \mid \bar{T}(z, z') \neq \emptyset & \sup \bar{T}(z, z') \geq t \}.
\]

\( X_{OA}(z, t) \) is the set of states that are output-adjacent to some state in \( z \) and can be reached from \( z \) via a path that takes at least \( t \) clock ticks and along which the output does not change, except at the final state. According to the above definition, in the update process after the \( t \)-th clock tick, those transitions in the diagnoser that could have only taken place before the \( t \)-th tick of the clock are ruled out. For example, for the diagnoser of Fig. 3.12, \( X_{OA}(z, t) = z'_1 \cup z'_2 \cup z'_3 \) for \( 0 \leq t \leq 3 \), \( X_{OA}(z, t) = z'_2 \cup z'_3 \) for \( 4 \leq t \leq 5 \), and \( X_{OA}(z, t) = z'_3 \) for \( t \geq 6 \).

Naturally, \( \kappa(\text{Pred}(z, t)) \) will be the prediction of the system's condition. Now suppose the failures are permanent and, hence, once a failure occurs, it will remain permanently. Then we can expect to be able to use the predictions for fault detection and isolation. Later in this section, in Theorems 3.6 and 3.7, we will establish a relation between \( \kappa(\text{Pred}(z_k, t)) \) and the condition estimates provided by the standard diagnoser. Before introducing the theorems, we discuss the construction of \( D_{\text{std}} \) in more detail and derive some related results which we need for the proof of Theorems 3.6 and 3.7.

\( G_\tau \) is the TDES model describing the system, with \( Q_\tau \) being its state set. In
order to design the standard diagnoser, first we incorporate the information about the occurrence of clock ticks in the output map using the method described in Section 2.1. Let \( G' \) be the resulting TDES with \( Q'_r (\supseteq Q_r \supseteq X) \) and \( Y' (= Y \times \{0,1\}) \), \( \lambda'_r : Q'_r \to Y' \), \( \kappa'_r : Q'_r \to \mathcal{K} \) as the corresponding state set, output set, output map and condition map. Every \( \tau \) transition \( q_1 \xrightarrow{\tau} q_2 \) in \( G_r \) is replaced with two transitions \( q_1 \xrightarrow{\tau} q'_1 \xrightarrow{\tau} q_2 \) in \( G'_r \) during which the output changes from \( (\lambda_r(q_1),0) \) to \( (\lambda_r(q_1),1) \), and to \( (\lambda_r(q_1),0) \) (By assumption, \( \lambda_r(q_1) = \lambda_r(q_2) \)). So in \( G'_r \), the clock tick is represented by two transitions. Let us refer to these back-to-back transitions as **complex \( \tau \) transitions**. Note that \( q_2 \), the state of \( G'_r \) after the complex \( \tau \) transition, is in \( Q_r \) (the state set of \( G_r \)). With the exception of the complex \( \tau \) transitions, the transition graphs of \( G_r \) and \( G'_r \) are identical.

The standard diagnoser, \( D_{std} \), is designed based on \( G'_r \). Let \( z_{std,k} \) denote the state estimate provided by \( D_{std} \) immediately after observing \( y_k \).

We assume that the initial state of \( D_{std} \) is given by

\[
 z_{std,0} = z_0 \cup \{ q \in Q_r \mid q \text{ is reachable from some } x \in z_0 \subseteq X \subseteq Q_r \text{ through} \]

a sequence of complex \( \tau \) transitions only}, \( \quad (3.1) \)

where \( z_0 \) is the initial state estimate of the predicting diagnoser. This reflects the assumption that at the time when the diagnosers are initialized, no information about the time of generation of the latest event and output is available. \(^2\) Note that \( z_{std,0} = z_{r,0} \), the initial estimate for the state of the TDES.

Since \( z_0 \subseteq X \), \( \lambda_r^{-1}(\{(y_1,0)\}) \cap z_0 = \lambda_r^{-1}(\{y_1\}) \cap z_0 = z_1 \) (\( z_1 \) is the state estimate provided by \( D \) after \( y_1 \) is read). Therefore

\[
 z_{std,1} = z_{std,0} \cap \lambda_r^{-1}(\{(y_1,0)\}) \\
 = (z_0 \cap \lambda_r^{-1}(\{y_1\})) \cup \{ q \in Q_r \mid q \text{ is reachable from some} \]

\[ x \in z_0 \cap \lambda_r^{-1}(\{y_1\}) \subseteq X \subseteq Q_r \text{ through a sequence of complex } \tau \]

\(^2\)We have not included any states from \( Q'_r - Q_r \) in \( z_{std,0} \) because they are fictitious states and provide no extra information about the state of the TDES.
transitions only} \]
= \{ q \in Q_\tau \mid q \textrm{ is reachable from some } x \in z_1 \subseteq X \subseteq Q_\tau \textrm{ through a sequence of complex } \tau \textrm{ transitions only}\}. \quad (3.2)

Note that \( z_{\text{std},1} = z_{\tau,1} \subseteq Q_\tau \) (\( z_{\tau,1} \) was defined in Sec. 3.2 as the estimate for the state of the TDES following the reading of the first output symbol, \( y_1 \)).

We will find the following version of output-adjacency in \( G'_\tau \) very useful.

**Definition 3.3** Suppose \( q, q' \in Q_\tau(\subseteq Q'_\tau) \) and \( \lambda_\tau(q) \neq \lambda_\tau(q') \). Then we say \( q' \) is \( \lambda_\tau \)-output-adjacent to \( q \) (in \( G'_\tau \)) and write \( q \Rightarrow_\lambda q' \) if there exists a path from \( q \) to \( q' \) in \( G'_\tau \) over which the first component of \( \lambda'_\tau \) remains constant except at the final transition (i.e., no output symbol from \( Y \) is generated except at the last transition).

\[ \square \]

In the above definition, the path may include complex \( \tau \) transitions. Also, since the last event causes the generation of a new output symbol from \( Y \), it must be a non-tick event. Thus, \( q' \in X \). Note that for \( q, q' \in Q \), \( q \Rightarrow q' \) in \( G_\tau \) if and only if \( q \Rightarrow_\lambda q' \) in \( G'_\tau \).

The following lemma establishes the relation between the state estimates provided by \( D_{\text{std}} \) and \( D \) following the generation of each new output symbol for \( k \geq 2 \).

**Lemma 3.3** Let \( z_{\text{std},0} \) be given by Eq.(3.1). Then we have

\[ z_{\text{std},k} = z_k, \quad k \geq 2. \]

**Proof.** First we show \( z_{\text{std},2} = z_2 \). Suppose \( x \in z_2 \subseteq X \). Then there exists \( q \in z_{\tau,1} \) and a path in \( G_\tau \) from \( q \) to \( x \), containing \( t_2 \) tick transitions, along which the output is \( \lambda_\tau(q) = y_1 \). Therefore, in \( G'_\tau \), \( x \) is reachable from \( q \in z_{\text{std},1} = z_{\tau,1} \) via a path, containing \( t_2 \) complex \( \tau \) transitions, along which no new output from \( Y \) is generated, except at the final transition (\( t_2 \) is the number of clock ticks between the reading of \( y_1 \) and the generation of \( y_2 \) in the output sequence of the system). Hence, \( x \in z_{\text{std},2} \). This shows \( z_2 \subseteq z_{\text{std},2} \). Similarly, we can show that \( z_{\text{std},2} \subseteq z_2 \).
Suppose $z_{\text{std},k} = z_k$. If $q \in z_{\text{std},k+1}$, then $q' \Rightarrow_{\lambda_r} q$, for some $q' \in z_{\text{std},k} \subseteq X$. There exists a path $P'$ from $q'$ to $q$ in $G'_r$, containing $t_{k+1}$ complex $\tau$ transitions, on which no new output symbol from $Y$ is generated except at the final transition. Also we have $q \in X$. After projecting out the complex $\tau$ transitions from $P'$, we arrive at a path from $q'$ to $q$ in $G$. Thus, in $G$, $q' \Rightarrow q$ and $t_{k+1} \in \tilde{T}(q', q)$; hence, $q \in z_{k+1}$. This shows that $z_{\text{std},k+1} \subseteq z_{k+1}$.

Conversely, if $x \in z_{k+1}$, then there exists $x' \in z_k = z_{\text{std},k}$, with $t_{k+1} \in \tilde{T}(x', x)$, and a path $P$ from $x'$ to $x$ on which the output does not change except at the final transition. There exists a (not necessarily unique) path $P'$ in $G'_r$, connecting $x'$ to $x$ such that $P'$ includes $t_{k+1}$ complex $\tau$ transitions, and by projecting out these complex $\tau$ transitions, we will get $P$ back. Hence, in $G'_r$, $x' \Rightarrow_{\lambda_r} x$ and $x \in z_{\text{std},k+1}$. □

For example, let $G_r$ be the simple TDES of Example 3.1. The TDES $G'_r$ is depicted in Fig. 3.13. In $G'_r$, $z_{\text{std},0} = z_{r,0} = \{0, 1, 2, 3, 4, 5\}$. Suppose $y_1 = 01$. Then $z_1 = \{0, 2\}$, $z_{\text{std},1} = z_{r,1} = \{0, 1, 2, 3\}$. Suppose a clock tick occurs. Then the state estimate provided by $D_{\text{std}}$ after the clock tick, denoted here by $z_{\text{std},1}(1)$, will be $z_{\text{std},1}(1) = \{1, 3\}$. If following this, $y_2 = 02$ is generated, then $D_{\text{std}}$ will update $z_{\text{std},1}(1)$ to $z_{\text{std},2} = \{4\}$ which equals $z_2$. If after this, a tick occurs, then $D_{\text{std}}$ will update its state estimate to $z_{\text{std},2}(1) = \{5\}$. Following this, the observation of $y_3 = 01$
will result in the generation of the estimate \( z_{\text{std}, 3} = \{0\} \) which is identical to \( z_3 \).

Now define \( \text{Adv} : 2^{Q_r} \times \mathbb{N} \rightarrow 2^{Q_r} \) as follows. For \( z \subseteq Q_r(\subseteq Q'_r) \),

- \( \text{Adv}(z, 0) := z \);
- \( \text{Adv}(z, 1) := \{ \text{The set of states in } Q_r(\subseteq Q'_r) \text{ that can be reached from a state in } z \text{ via a path in } G_r \text{ such that no new output symbol is generated on the path (i.e., the first component of } \lambda'_r \text{ remains constant) and the path consists of } n \text{ non–tick transitions } (n \geq 0), \text{ followed by a complex } \tau \text{ transition} \}; \)
- \( \text{Adv}(z, t) := \text{Adv}(\text{Adv}(z, t-1), 1), \text{ for } t \geq 2. \)

For \( z \subseteq Q_r \), \( \text{Adv}(z, t) \) is the set of states that the TDES can reach from \( z \) in \( t \) clock ticks. Observe that \( \text{Adv}(z_{\text{std}, 1}, t) \) is the state estimate provided by \( D_{\text{std}} \) after \( t \) ticks before \( y_2 \) is generated. Since output changes are the results of non–tick events, \( z_{\text{std}, 2} \subseteq X \subseteq Q_r \). Similarly, \( z_{\text{std}, k} \subseteq X \subseteq Q_r \) for \( k > 2 \). Therefore, if \( z_{\text{std}, k}(t) \) denotes the state estimate provided by \( D_{\text{std}} \) after \( t \) clock ticks following the generation of \( y_k \), before the (possible) occurrence of \( y_{k+1} \), then \( z_{\text{std}, k}(t) = \text{Adv}(z_{\text{std}, k}, t) \).

Define \( Q_{\text{OA}} : 2^{Q_r} \times \mathbb{N} \rightarrow 2^{Q_r} \) as follows. For \( z \subseteq Q_r \subseteq Q'_r \)

\[
Q_{\text{OA}}(z, t) := \{ q \in Q_r \mid \exists q' \in \text{Adv}(z, t) \& q' \Rightarrow_{\lambda_r} q \}.
\]

\( Q_{\text{OA}}(z, t) \) is the set of states that are \( \lambda_r \)-output–adjacent to some state in \( \text{Adv}(z, t) \). Note that since the states in \( Q_{\text{OA}}(z, t) \) can be reached by non–tick events (recall that only non–tick events result in output change), \( Q_{\text{OA}}(z, t) \subseteq X \).

\( Q_{\text{OA}}(z, t) \) can be interpreted as follows. \( q \in Q_{\text{OA}}(z, t) \subseteq X \) if it can be reached from a state \( q' \in z \) via a path such that on the path no new output symbol from \( Y \) is generated except at the final state and the path includes at least \( t \) complex \( \tau \) transitions. For instance, in the simple TDES of Example 3.1, \( \text{Adv}({4, 9}, 0) = \{4, 9\}, \text{Adv}({4, 9}, 1) = \{5\}, Q_{\text{OA}}({4, 9}, 0) = \{0, 6\}, \text{and } Q_{\text{OA}}({4, 9}, 1) = \{0\}. \)

**Lemma 3.4** If \( z \in Z - \{z_0\} \), then \( X_{\text{OA}}(z, t) = Q_{\text{OA}}(z, t) \).
Proof. Suppose $q \in Q_{OA}(z, t) \subseteq X$. Then it can be reached from a state $q' \in z$ via a path such that on the path no new output symbol from $Y$ is generated, except at the final transition, and the path includes at least $t$ complex $\tau$ transitions. Therefore, since $q' \in z \subseteq X$, $t \leq \sup \bar{\tau}(q', q)$. Conversely, if for some $q' \in z \subseteq X$ and $q \in X$, $\bar{\tau}(q', q) \neq \emptyset$ and $t \leq \sup \bar{\tau}(q', q)$, then $q \in Q_{OA}(z, t)$. As a result,

$$Q_{OA}(z, t) = \{ q \in X \subseteq Q_{\tau} | \exists q' \in z : \bar{\tau}(q', q) \neq \emptyset \& \sup \bar{\tau}(q', q) \geq t \}$$

$$= \cup \{ z' \mid \bar{\tau}(z, z') \neq \emptyset \& \sup \bar{\tau}(z, z') \geq t \}$$

$$= X_{OA}(z, t).$$

\[\square\]

For a diagnoser estimate $z$, as explained before, $X_{OA}(z, t)$, is the set of states in $G$ that are output-adjacent (in $G$) to some state in $z$ and can be reached via a path which contains at least $t$ clock ticks. On the other hand, $Q_{OA}(z, t)$ is the set of states in $G'_{\tau}$ that are $\lambda_{\tau}$-output-adjacent (in $G'_{\tau}$) to some state in $z$ and can be reached via a path in $G'_{\tau}$ containing at least $t$ complex $\tau$ transitions. The result $X_{OA}(z, t) = Q_{OA}(z, t)$ is to be expected since for two states $x, x' \in X$, $x \Rightarrow x'$ in $G$ if and only if $x \Rightarrow \lambda_{\tau} x'$ in $G'_{\tau}$ (by Lemma 3.1), hence if and only if $x \Rightarrow x'$ in $G'_{\tau}$. For instance, in Example 3.1, for $z = (\{4, 9\}, 2)$, $X_{OA}(z, 0) = \{0, 6\} = Q_{OA}(\{4, 9\}, 0)$, $X_{OA}(z, 1) = \{0\} = Q_{OA}(\{4, 9\}, 1)$, and $X_{OA}(z, 2) = \emptyset = Q_{OA}(\{4, 9\}, 2)$ (Refer to Figures 3.10 and 3.13).

Lemma 3.5 For all $t \geq 0$, $Q_{OA}(z_{std, 1}, t) = Q_{OA}(z_1, t)$.

Proof. $Q_{OA}(z_1, t) \subseteq Q_{OA}(z_{std, 1}, t)$ since $z_1 \subseteq z_{std, 1}$. Suppose $q \in Q_{OA}(z_{std, 1}, t)$. Then $q$ can be reached from some $q' \in z_{std, 1}$ in $t'$ clock ticks (for some $t' \geq t$), via a path along which no output symbol from $Y$ is generated (except at the final state). $q'$, in turn, can be reached from some $q'' \in z_1$ on a path consisting of complex $\tau$ transitions only. Along this path, no new symbol from $Y$ is generated. Therefore, $q$ is reachable from $q'' \in z_1$ in $t''$ clock ticks (for some $t'' \geq t' \geq t$) via a path along which no new symbol from $Y$ is generated. Thus, $q \in Q_{OA}(z_1, t)$.\[\square\]
Theorem 3.6 For a permanent failure mode $F_i$, after $t$ ticks of the clock following the occurrence of the observation $y_k$, and before the (possible) generation of $y_{k+1}$, if the standard diagnoser, $D_{std}$, is in an $F_i$-certain state, then $\text{Pred}(z_k, t)$ will also be $F_i$-certain, where $z_k$ is the state of the predicting diagnoser $D$ upon observing $y_k$.

Proof. In order to prove the theorem, we first define predictions of the states of $G'_r$ based on the estimates provided by $D_{std}$. Then we show that the condition estimates based on these predictions are identical to those obtained from the predictions provided by the predicting diagnoser. It follows that if $D_{std}$ is in an $F_i$-certain state, then the predictions provided by $D_{std}$, and hence, those given by $D$ will be $F_i$-certain.

$z_{std,k}$ is the state estimate of $D_{std}$ following the generation of $y_k$. Let $z_{std,k}(t)$ denote the state estimate after $t$ clock ticks, with $0 \leq t \leq t_{k+1}$, before the (possible) occurrence of $y_{k+1}$. This definition implies $z_{std,k} = z_{std,k}(0)$. For any state of $D_{std}$, $z_{std}$, define

$$Rch(z_{std}) := \{q \in Q'_r \mid q \text{ is reachable from a state in } z_{std} \text{ via a path along which no new output symbol from } Y \text{ is generated}\}$$

$$= \{q \in Q'_r \mid q \text{ is reachable from a state in } z_{std} \text{ via a path along which the first component of the output } \lambda'_r(\cdot) \text{ remains the same}\}.$$ 

Also let

$$\text{Lim}(z_{std}) := \{q \in Q'_r \mid q \text{ belongs to a cycle in } Rch(z_{std})\}.$$ 

$\text{Lim}(z_{std})$ is the set of cycles in $Rch(z_{std})$ in which the system can be trapped for an arbitrarily long time without generating a new output symbol from $Y$. (Note that in $G_r$ and hence in $G'_r$, the unfolding of events never stops since at least the tick event happens periodically. Thus, $G_r$ (and $G'_r$) do not have any deadlock states; i.e., for any $q \in Q_r$, $\delta_r(q, \sigma) \neq \emptyset$, for some $\sigma \in \Sigma_r$).

Similar to $D$, define the prediction after $t$ ticks following $y_k$ provided by $D_{std}$
according to

\[ \text{Pred}(z_{\text{std,k}}(t)) = \lim(z_{\text{std,k}}(t)) \cup \{ q \mid \exists q' \in z_{\text{std,k}}(t) \& q' \Rightarrow_{\lambda} q \} \]
\[ = \lim(z_{\text{std,k}}(t)) \cup Q_{\text{OA}}(z_{\text{std,k}}, t). \]

\text{Pred}(z_{\text{std,k}}(t)) \) can be interpreted as follows. Following the \( t \)-th tick, either a new output symbol from \( Y \) \( (y_{k+1}) \) will be generated after some time (and the state of \( G'_r \) enters \( Q_{\text{OA}}(z_{\text{std,k}}, t) \)), or no new output is generated and the state of \( G'_r \) will enter \( \lim(z_{\text{std,k}}(t)) \) after a finite time and stay there for all future time.

We want to show that \( \kappa(\text{Pred}(z_k, t)) = \kappa'_r(\text{Pred}(z_{\text{std,k}}(t))) \). For this, we prove that
(i) \( \lim(z_{\text{std,k}}(t)) = \lim(z_{\text{std,k}}) \), and (ii) \( \kappa(\lim(z_k)) = \kappa'_r(\lim(z_{\text{std,k}})) \).

(i) \( \lim(z_{\text{std,k}}(t)) = \lim(z_{\text{std,k}}(0)) = \lim(z_{\text{std,k}}) \) because \( \lim(z_{\text{std,k}}(t)) \) is the set of cycles in \( \text{Rch}(z_{\text{std,k}}) \) that can be reached from \( z_{\text{std,k}} \) in \( t \) clock ticks or more. This includes all of the cycles in \( \text{Rch}(z_{\text{std,k}}) \) because if a cycle can be reached in \( t'(< t) \) ticks, then since the system can stay in the cycle for an arbitrarily long time, we can think of the cycle as one reachable in \( t'' > t \) clock ticks, for some \( t'' \geq t \).

(ii) Next, we prove that \( \kappa(\lim(z_k)) = \kappa'_r(\lim(z_{\text{std,k}})) \).

\( \kappa(\lim(z_k)) \subseteq \kappa'_r(\lim(z_{\text{std,k}})) \) because for every state or cycle in \( \lim(z_k) \), there exists a cycle in \( \lim(z_{\text{std,k}}) \) having the same condition as shown below.

- For every cycle \( C \) in \( \text{Rch}(z_k) \) (i.e., \( C \subseteq \lim(z) \)), there is a cycle \( C'_r \) in \( \text{Rch}(z_{\text{std,k}}) \) consisting of the same transitions in \( C \) plus some complex \( \tau \) transitions such that by projecting out the complex \( \tau \) transitions in \( C'_r \), we can get \( C \) back. Since the condition of the system does not change as a result of \( \tau \) transition (the clock tick), \( \kappa(C) = \kappa'_r(C'_r) \).

- Assume \( \lim_1(z_k) \neq \emptyset \) and let \( x^* \in \lim_1(z_k) \). \( x^* \in \text{Rch}(z_{\text{std,k}}) \) since \( z_k \subseteq z_{\text{std,k}} \). Suppose the state of \( G'_r \) reaches \( x^* \). From this point on, only \( \tau \) transitions will occur in \( G'_r \). This means that there exists a cycle \( C_{x^*} \) in \( G'_r \) consisting of \( \tau \) events only, and reachable from \( x^* \) through \( \tau \) events. Hence \( \kappa(x^*) = \kappa'_r(C_{x^*}) \).

As a result, there exists a cycle \( C'_{x^*} \) consisting of complex \( \tau \) transitions in \( G'_r \) in \( \text{Rch}(z_{\text{std,k}}) \) such that \( \kappa(x^*) = \kappa'_r(C'_{x^*}) \).
Similarly, if \( \text{Lim}_2(z_k) \neq \emptyset \), then for every \( x^* \in \text{Lim}_2(z_k) \), \( G'_r \) contains a cycle \( C'_{x^*} \) in \( \text{Rch}(z_{\text{std},k}) \) with \( \kappa(x^*) = \kappa'_r(C'_{x^*}) \).

To show \( \kappa'_r(\text{Lim}(z_{\text{std},k})) \subseteq \kappa(\text{Lim}(z_k)) \), we show that for every cycle in \( \text{Lim}(z_{\text{std},k}) \), there exists a state or cycle in \( \text{Lim}(z_k) \) with the same condition. There are two cases.

**Case 1.** If \( C'_r \) is a cycle of \( G'_r \) in \( \text{Rch}(z_{\text{std},k}) \) corresponding to a cycle \( C_r \) in \( G_r \) which contains non-tick events, then, after projecting out \( \tau \) events, we will get a cycle \( C \) of \( G \) in \( \text{Rch}(z_k) \), with \( \kappa'_r(C'_r) = \kappa_r(C_r) = \kappa(C) \) (See Fig. 3.14 for an example).

**Case 2.** Suppose \( C'_r \) is a cycle of \( G'_r \) in \( \text{Rch}(z_{\text{std},k}) \), consisting of complex \( \tau \) transitions only. It follows from Lemma 3.3 and Eq.(3.2) that any state in \( \text{Rch}(z_{\text{std},k}) \) is reachable via a path (in \( G'_r \)) starting in \( z_k \), lying entirely in \( \text{Rch}(z_{\text{std},k}) \). Let \( P' \) be a path starting at some \( x \in z_k \) which ends at a state belonging to \( C'_r \). If all of the transitions in \( P \) are complex \( \tau \) transitions, then \( x \in \text{Lim}(z_k) \) and \( \kappa(x) = \kappa'_r(P' \cup C'_r) = \kappa'_r(C'_r) \). If \( P' \) includes non-tick events, then let \( x^* \) be the state following the last non-tick transition in \( P' \) (See Fig. 3.15 for an example).

Obviously \( x^* \in X \). \( C'_r \) is reachable from \( x^* \) through a sequence of complex \( \tau \) transitions. Hence, \( x^* \in \text{Lim}(z_k) \) and \( \kappa(x^*) = \kappa'_r(C'_r) \).

This shows that \( \kappa'_r(\text{Lim}(z_{\text{std},k})) \subseteq \kappa(\text{Lim}(z_k)) \). Thus,

\[
\kappa(\text{Lim}(z_k)) = \kappa'_r(\text{Lim}(z_{\text{std},k})).
\] (3.3)
Figure 3.13: Proof of Thm. 3.6. Case 2. \( \tau' \) stands for complex \( \tau \) transition and \( \sigma \neq \tau' \).

Now observe that by Lemmas 3.4, 3.5 and 3.3,

\[
Q_{OA}(z_{std,k},t) = Q_{OA}(z_k,t) = X_{OA}(\bar{z}_k,t). \tag{3.4}
\]

From Eqs.(3.3) and (3.4), it follows that

\[
\kappa(\text{Pred}(\bar{z}_k,t)) = \kappa'_r(\text{Pred}(z_{std,k}(t))). \tag{3.5}
\]

Therefore the predictions of condition based on the models \( G'_r \) and \( G \) are identical. This should not come as a surprise because both models describe the same system with the same accuracy (for the purpose of estimating the condition of the system).

If \( z_{std,k}(t) \) is \( F_i \)-certain, then \( \text{Pred}(z_{std,k}(t)) \) will be \( F_i \)-certain (since \( F_i \) is assumed permanent). As a result, by Eq.(3.5), \( \text{Pred}(\bar{z}_k,t) \) will be \( F_i \)-certain too. \( \square \)

The following theorem proves that if the predicting diagnoser \( D \) detects and isolates a failure mode \( F_i \), then the standard diagnoser will do so with a delay shorter than \( |Q_r| \) clock ticks.

**Theorem 3.7** Suppose \( F_i \) is a permanent failure mode. If at some point in time, the state of the predicting diagnoser or its prediction are \( F_i \)-certain, then the standard diagnoser, \( D_{std} \), will enter an \( F_i \)-certain state in less than \( |Q_r| \) clock ticks.

**Proof.** If after the generation of the output \( y_k \in Y \), \( \bar{z}_k \) becomes \( F_i \)-certain, then \( z_{std,k} \) will be \( F_i \)-certain (since \( \kappa'_r(z_{std,k}) = \kappa(z_k) \)) and as a result, \( F_i \) is detected
and isolated by $D_{\text{std}}$. If in $t$ clock ticks $(0 \leq t \leq t_{k+1})$ following the generation of $y_k$, $\text{Pred}(z_k, t)$ becomes $F_1$-certain, then, by Theorem 3.6, so will be $\text{Pred}(z_{\text{std}, k}(t))$. Therefore, $\text{Lim}(z_{\text{std}, k}(t))$ and $Q_{\text{OA}}(z_{\text{std}, k}, t)$ will be $F_1$-certain. Note that $Q_{\text{OA}}(z_{\text{std}, k}(t))$ is the set of states in $Q_r'$ that are $\lambda_r$-output-adjacent to some state in $z_{\text{std}, k}(t)$. Let $q_1q_2\cdots$ be an arbitrary infinite path in $G_r'$ starting in $z_{\text{std}, k}(t)$, i.e., $q_1 \in z_{\text{std}, k}(t)$. (Note that $G_r'$ contains no deadlock states; i.e., for any state $q \in Q_r'$, there exists $\sigma \in \Sigma_r$ such that $\delta(q, \sigma) \neq \emptyset$, because at least the clock always ticks). If $\{q_1\}$ is $F_1$-certain, then so will be $\{q_i\}$, for all $i \geq 2$, because the failure modes are assumed permanent. If $\{q_1\}$ is not $F_1$-certain, then there exists $n$, with $1 < n \leq |Q_r'|$ such that $q_n \in \text{Lim}(z_{\text{std}, k}(t))$ or $q_n \in Q_{\text{OA}}(z_{\text{std}, k}, t)$ and, hence, $\{q_n\}$ will be $F_1$-certain which in turn implies that $\{q_i\}$ is $F_1$-certain for $i \geq n$. We showed that on any path $q_1q_2\cdots$ starting in $z_{\text{std}, k}(t)$, $\{q_i\}$ is $F_1$-certain for $i \geq |Q_r'|$. A path in $G_r'$ consisting of $|Q_r'| - 1$ transitions can contain at most $(|Q_r'| - 1)/2 \leq |Q_r| - 1$ complex $\tau$ transitions. Therefore in less than $|Q_r|$ clock ticks following the generation of the estimate $z_{\text{std}, k}(t)$ (by $D_{\text{std}}$), whether a new output symbol is generated in between the clock ticks or not, the state estimate provided by $D_{\text{std}}$ will be $F_1$-certain. \[\square\]

Theorems 3.6 and 3.7 establish that the predicting diagnoser $D$ is at least as fast as the standard diagnoser $D_{\text{std}}$ in detecting and isolating failures. In some cases, $D$ could even be faster than the standard diagnoser. For instance, if a failure $F_2$ is caused by another failure $F_1$, then the predicting diagnoser may predict $F_2$ (when it detects $F_1$) even before $F_2$ occurs. Obviously, the standard diagnoser cannot detect $F_2$ before it happens.

In summary, the diagnoser provides an estimate of the system's condition ($\kappa(z_k)$) every time a new output symbol is generated. Between consecutive output symbols, the predictions $\kappa(\text{Pred}(z_k, t))$ can be used for diagnosing permanent failures. Going back to the diagnoser for the neutralization process (Fig. 3.11), we can easily verify that for $z = \{1, 1''\}$, $\text{Lim}(z) = \{1'', 1.1''\}$, and

\[
\text{Pred}(z, t) = \{3, 3', 4, 4'\} \cup \{1'', 1.1''\} \quad \text{for} \quad 0 \leq t \leq 2,
\]

\[
\text{Pred}(z, t) = \{1'', 1.1''\} \quad \text{for} \quad t \geq 3.
\]

86
Therefore when the diagnoser is at $z = \{1, 1''\}$, if no new output symbol is generated for three ticks of the clock, then it can be concluded that $F_2$ has occurred. This deduction is shown in Fig. 3.11 by a transition (depicted by a dashed line) from $z = \{1, 1''\}$ to $\text{Pred}(z, t) = \{1'', 1.1''\}$. Similarly, it can be seen that $\text{Lim}(\{1, 1'', 1.1'', 2, 2.1', 2.2'\}) = \{1'', 1.1''\}$ and $\text{Lim}(\{1.7'\}) = \{1.7'\}$. For any other state estimate $z$, $\text{Lim}(z) = \emptyset$.

The approach presented in this chapter may result in reduction in on-line computations and in the size of the diagnoser.

In the predicting diagnoser, no update of the estimate of the system's state is required at clock ticks. Between clock ticks, predictions can be used for fault diagnosis. In many cases, however, there is no need for updating condition estimates between consecutive outputs, and estimates at the instants of generation of new output symbols are sufficient for fault diagnosis. For instance, in the neutralization process, there is no need for using predictions between consecutive output symbols, except when the output is $(C1, L0, n)$ (and only after the third clock tick). For example, suppose the diagnoser (Fig. 3.11) is in the state $z = (\{5, 5'\}, 2)$. After the diagnoser enters $z$, there is no need for estimate update till the next output is generated at which point, one needs to make sure that the output and its timing are valid and acceptable (Certainly if no new output symbol is generated before the $u + 1$– the tick occurs, then there will be an inconsistency between the system's model and the observed behaviour). In these cases, our approach can reduce the on-line computations significantly because it does not require estimate updates at clock ticks.

In many cases, the predicting diagnoser may have significantly fewer states than the standard diagnoser. This usually happens when the time bounds of events are large numbers. For example, in the neutralization process, filling the tank takes a lot longer than opening a valve (assumed here to take 1 clock tick). Therefore $u$ and $u'$ are considerably larger than 1. If we assume $u = 200$ and $u' = 100$, then the TDES will have about 1700 states. As a result, while both the standard diagnoser and the diagnoser based on the methodology of [4] will have at least a few hundred states, the predicting diagnoser (Fig. 3.11) has only 19 states and, except for some of the
transition–time sets \((\tilde{T}(.,.))\), the structure of our diagnoser does not depend on the parameters \(l, u, l'\) and \(u'\). In general, a small clock period is necessary for accurate time keeping in a TDES. If the system contains slow and fast processes, then the clock period has to be sufficiently small for accurate modelling of the fast processes. This results in large time bounds for the events of the slow processes, hence, a large state set for the TDES. In these cases, our approach can lead to significant reduction in the size of the diagnoser. The simple TDES of Example 3.1, provides an example of a case where the sizes of the predicting and standard diagnosers are close (This is not difficult to verify). In this example, each event is followed by another event in at most 2 clock ticks.

The above improvements are obtained at the expense of more off-line design calculations, in particular, the computation of the transition–time sets \(T\) and \(\tilde{T}\) (Refer to Remarks 3.2 and 3.3).

Finally, we note that for any subset \(z\) generated by the diagnoser, \(\text{Lim}(z)\) needs to be calculated. For this, it is computationally economical to compute \(\text{Lim}(\{x\})\) for all \(x \in X\) since \(\text{Lim}(z) = \bigcup\{\text{Lim}(\{x\}) \mid x \in z\}\). \(\text{Lim}(\{x\})\) can be computed in polynomial time.

### 3.3 Time–Diagnosability

In this section, the issue of time–diagnosability of failures will be studied. We will assume that failures are permanent and will only discuss single–failure scenarios. Extension of our results to the case of simultaneous failures is similar to that in Section 2.5 for finite–state automata.

First, we obtain a necessary and sufficient condition for time–diagnosability in terms of the standard diagnoser. Next, we will derive another set of necessary and sufficient conditions in terms of the predicting diagnoser.

We begin with the definition of time–diagnosability.

**Definition 3.4** A permanent failure mode \(F_i\) is **time–diagnosable** if there exists an integer \(T_i \geq 0\) such that following both the occurrence of the failure and initialization
of the standard diagnoser, $F_i$ can be detected and isolated after the occurrence of at most $T_i$ ticks.

Note that unlike the event-based version [4], in our definition of time-diagnosability, no assumption is made about the system's state or condition at the time the diagnoser is started.

**Remark 3.4** In Def. 3.4, time-diagnosability is defined with respect to the standard diagnoser $D$. It follows from Theorems 3.6 and 3.7 that a failure is time-diagnosable (with respect to $D_{std}$) if and only if it is time-diagnosable with respect to the predicting diagnoser $D$, i.e., it can be detected and isolated by the predicting diagnoser in less than a bounded number of clock ticks.

**Remark 3.5** If the TDES model of the system is activity-loop-free, then the notions of diagnosability (Section 2.3) and time-diagnosability will become equivalent for the TDES. This is because the number of non-tick events between two consecutive clock ticks will be bounded by $|Q_r| - 1$, where $|Q_r|$ is the cardinality of $Q_r$.

**Necessary and sufficient condition for time-diagnosability in terms of the standard diagnoser**

As mentioned in the previous section, in order to design the standard diagnoser, the information about the clock ticks has to be transferred and included in the output map of the TDES. Suppose $G_r' = (Q_r', \Sigma_r', \delta_r', q_0', Y_r', \lambda_r')$ is the resulting TDES and $\kappa_r' : Q_r' \to K$ the corresponding condition map. The state set $Q_r'$ can be partitioned according to the system's condition: $Q_r' = Q_{r,N} \cup Q_{r,F_1} \cup \cdots \cup Q_{r,F_p}$. Let $z$ be an estimate of the state of $G_r'$ given by the standard diagnoser. In a single-failure scenario, $z$ is $F_i$-certain if and only if $\kappa_r'(z) = \{F_i\}$, $F_i$-uncertain if and only if $F_i \in \kappa_r'(z)$ and $\kappa_r'(z) \neq \{F_i\}$, and an output symbol $y \in Y_r'$ is $F_i$-indicative if and only if $\lambda_r'^{-1}(\{y\}) \subseteq Q_{r,F_i}$.

It follows from Theorem 2.1 that assuming single-failure scenario and $z_0 = Q_{r,F_1}$, a permanent failure mode $F_i$ is time-diagnosable if and only if (i) from every $q \in Q_{r,F_i}$, there is (at least) one transition to another state in $Q_{r,F_i}$ unless $\lambda_r'(q)$ is $F_i$-indicative;
(ii) there is no cycle in $Q'_{r,F_i}$ consisting of states having the same output unless the output symbol is $F_i$-indicative; and (iii) there are no $F_i$-indeterminate cycles in the standard diagnoser.

Condition (i) is always satisfied because the clock always ticks. Condition (ii) always holds too because the activity-loop-free assumption implies that there must be a clock tick (hence, change of output) in every cycle in the TDES. Therefore we have the following result.

Theorem 3.8 Assume single-failure scenario and $z_0 = Q'_{r}$. A permanent failure $F_i$ is time-diagnosable if and only if there are no $F_i$-indeterminate cycles in the standard diagnoser for the TDES.

Necessary and sufficient conditions for time-diagnosability in terms of the predicting diagnoser

Now we study time-diagnosability in terms of the timed finite-state Moore automaton $G$ and the predicting diagnoser $D$. As mentioned in Section 3.2, the state set $X$ can be partitioned according to the system’s condition: $X = X_N \cup X_{F_1} \cup \cdots \cup X_{F_p}$ and this partition coincides with $\ker \kappa$. Let $z$ be an estimate of the state of $G$ given by the diagnoser. In a single-failure scenario, $z$ is $F_i$-certain if and only if $\kappa(z) = \{F_i\}$ and $F_i$-uncertain if and only if $F_i \in \kappa(z)$ and $\kappa(z) \neq \{F_i\}$. Furthermore, an output symbol $y \in Y$ is $F_i$-indicative if and only if $\lambda^{-1}(\{y\}) \subseteq X_{F_i}$.

The necessary and sufficient conditions for time-diagnosability in TDES resemble those for diagnosability in finite-state automata (Section 2.3). If in a failure mode, the TDES stops generating new output symbols, then it should be possible to determine the failure mode either from the last output symbol or from the time elapsed since the last output symbol was generated. Also, the diagnoser should not get trapped in a cycle of fault-uncertain states. For this, the diagnoser should not have fault-indeterminate cycles. The definition of fault-indeterminate cycles of the predicting diagnoser is very similar to that for the diagnosers for finite-state automata.

Definition 3.5 Suppose $z^1, \ldots, z^m$ is a cycle of $F_i$-uncertain states of the predicting diagnoser. The cycle is called $F_i$-indeterminate if there exist $l \geq 1$ and...
$x_1', x_2', \ldots, x_l' \in z'$, for all $1 \leq j \leq m$ such that $x_k^j \in X_{F_i}$ for all $1 \leq j \leq m$, $1 \leq k \leq l$ and $x_1', x_2', \ldots, x_l', \ldots, x_2^m, \ldots, x_l^m$ form a cycle in the timed RTS. The cycle in the timed RTS is called an **underlying faulty cycle** of the $F_i$-indeterminate cycle.

Let $e^i$ denote the second component of $z^i$ in the above definition, i.e., $z^i = (z^i, e^i)$. Then $e^i = 2$ because diagnoser states with $e = 1$ are reachable only from the initial state $z_0$ and $z_0$ is not reachable from any state.

Let us define:

\[
\begin{align*}
\text{Lim}(X_N) & := \bigcup \{\text{Lim}(\{x\}) \mid x \in X_N\}, \\
\text{Lim}(X_{F_i}) & := \bigcup \{\text{Lim}(\{x\}) \mid x \in X_{F_i}\}, \quad 1 \leq i \leq p \\
\lambda(\text{Lim}(X_{F_i})) & := \{\lambda(x) \mid x \in \text{Lim}(X_{F_i})\}, \quad 1 \leq i \leq p.
\end{align*}
\]

$\text{Lim}(\{x\})$ was previously defined in Section 3.2. $\text{Lim}(X_{F_i})$ (resp. $\text{Lim}(X_N) \cap X_N$) is the set of cycles (with constant output) and states in $X_{F_i}$ (resp. $X_N$) in which the system can get trapped for an arbitrarily long time and therefore generate no new output symbol.

**Theorem 3.9** Assume single-failure scenario and $z_0 = X$. A permanent failure $F_i$ is time–diagnosable if and only if

1. \[
\lambda^{-1}(\lambda(\text{Lim}(X_{F_i}))) \cap ((\text{Lim}(X_N) \cap X_N) \cup (\bigcup_{j \neq i} \text{Lim}(X_{F_j}))) = \emptyset; \quad (3.6)
\]

2. There are no $F_i$-indeterminate cycles in the predicting diagnoser $D$. \quad \Box

Before providing the proof, we explain the meaning of condition 1. Condition 1 can be rewritten as:

\[
\lambda^{-1}(\{y\}) \cap ((\text{Lim}(X_N) \cap X_N) \cup (\bigcup_{j \neq i} \text{Lim}(X_{F_j}))) = \emptyset, \quad \forall y \in \lambda(\text{Lim}(X_{F_i})). \quad (3.7)
\]
If $\text{Lim}(X_{F_i}) = \emptyset$, then Eq. (3.6) is satisfied. $\text{Lim}(X_{F_i}) = \emptyset$ means that after the occurrence of $F_i$, the system will continue generating new output symbols no later than every $t_{\text{max}}$ ticks of the clock (i.e., $t_k \leq t_{\text{max}}$) where $t_{\text{max}}$ is defined according to

$$t_{\text{max}} := \max_{x,x' \in X} \{ \max \tilde{T}(x,x') | \tilde{T}(x,x') \neq \emptyset \& \sup \tilde{T}(x,x') < \infty \}.$$ 

Certainly $t_{\text{max}} \leq |Q_r|$. 

Suppose $\text{Lim}(X_{F_i}) \neq \emptyset$ and $y \in \lambda(\text{Lim}(X_{F_i}))$ for some $y \in Y$. This means that after generating the output $y$, the system could delay generating the next output for an arbitrarily long time. But Eq. (3.7) implies that a delay longer than $t_{\text{max}} + 1$ ticks can only happen when the condition of the system is $F_i$. Hence, any delay longer than $t_{\text{max}} + 1$ ticks in generating a new output symbol while the output is $y$ implies that $F_i$ has occurred. 

In fault diagnosis in finite-state automata, we only rely on the output symbols for diagnosis and if in a failure condition $F_i$, the system stops generating new output symbols, the last output has to be $F_i$-indicative or $F_i$ will not be diagnosable. But for TDES, in addition to the output symbols, we also use the timing information which makes the diagnosis more accurate. 

We need the following lemmas to prove Theorem 3.9. In particular, Lemmas 3.11 to 3.14 explain the implications of condition 1 in more detail.

**Lemma 3.10** For $z \in Z - \{z_0\}$,

$$\bigcup \{z' | \tilde{T}(z,z') \neq \emptyset \& \sup \tilde{T}(z,z') \geq t_{\text{max}} + 1 \} = \bigcup \{z' | \tilde{T}(z,z') \neq \emptyset \& \sup \tilde{T}(z,z') = \infty \}.$$ 

**Proof.** Follows from:

$$\bigcup \{z' | \tilde{T}(z,z') \neq \emptyset \& \sup \tilde{T}(z,z') \geq t_{\text{max}} + 1 \} = \{x' | \exists x \in z : \tilde{T}(x,x') \neq \emptyset \& \sup \tilde{T}(x,x') \geq t_{\text{max}} + 1 \} = \{x' | \exists x \in z : \tilde{T}(x,x') \neq \emptyset \& \sup \tilde{T}(x,x') = \infty \} = \bigcup \{z' | \tilde{T}(z,z') \neq \emptyset \& \sup \tilde{T}(z,z') = \infty \}.$$
Lemma 3.11 Assume \( z \) is an estimate of the system's state provided by the predicting diagnoser. If \( \lim(z) \cap \lim(X_{F_i}) \neq \emptyset \) and Eq. (3.6) holds, then \( \lim(z) \subseteq \lim(X_{F_i}) \).

Proof. \( \lim(z) \subseteq \lim(X_N) \cup (\bigcup_{1 \leq j \leq p} \lim(X_{F_j})) = (\lim(X_N) \cap X_N) \cup (\bigcup_{1 \leq j \leq p} \lim(X_{F_j})) \).

Let \( x \in \lim(z) \cap \lim(X_{F_i}) \). If \( \lim(z) \not\subseteq \lim(X_{F_i}) \), then \( \lim(z) \cap ((\lim(X_N) \cap X_N) \cup (\bigcup_{j \neq i} \lim(X_{F_j}))) \neq \emptyset \). Let \( x' \in \lim(z) \cap ((\lim(X_N) \cap X_N) \cup (\bigcup_{j \neq i} \lim(X_{F_j}))) \). Since \( x' \in \lim(z) \), \( \lambda(x) = \lambda(x') \); hence \( x' \in \lambda^{-1}(\lambda(\lim(X_{F_i}))) \). This contradicts Eq. (3.6). \( \square \)

Lemma 3.12 Suppose \( \hat{T}(z, z') \neq \emptyset \) and \( \sup \hat{T}(z, z') = \infty \) for \( z, z' \in Z \) in the diagnoser. If \( \lim(z) \subseteq \lim(X_{F_i}) \), then \( \kappa(z') = \{ F_i \} \).

Proof. Since \( \hat{T}(z, z') \) is unbounded, for any \( z' \in z' \), there exists a path \( x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_l \rightarrow x' \) in \( G \) with \( x_i \in z, l \geq 1, \lambda(x_i) = \lambda(x_1) \) for \( 1 \leq i \leq l \), such that either it includes a cycle and/or at least one transition with an unbounded transition-time set \( \hat{T}(\cdot, \cdot) \). Hence, there exists \( k \), with \( 1 \leq k \leq l \) such that \( x_k \in \lim(z) \subseteq \lim(X_{F_i}) \subseteq X_{F_i} \) (i.e., the path goes through \( \lim(z) \subseteq \lim(X_{F_i}) \)). Since \( F_i \) is permanent, \( x' \in X_{F_i} \) and hence, \( \kappa(z') = \{ F_i \} \). \( \square \)

Lemma 3.13 Suppose the diagnoser reaches a state \( z = (z, e) \). Assume that (i) the failure event \( F_i \) occurred before the diagnoser entered \( z \), (ii) Eq. (3.6) holds, and (iii) no new output is generated after the diagnoser enters \( z \) for at least \( t_{\text{max}} + 1 \) clock ticks. Then

\[ \kappa(\text{Pred}(z, t_{\text{max}} + 1)) = \{ F_i \} . \]

Proof. First we show that there exists \( x \in z \cap X_{F_i} \) such that \( \lim(\{ x \}) \neq \emptyset \). There are two cases to examine.

- \( e = 1 \). In this case, \( z \) is the diagnoser state after the first output \( y_1 \) is read.

It follows from (i) that when \( y \) was read, the state of the TDES was some \( q \in z_{t_1} \cap Q_{t,F_i} \). \( q \) is reachable from some \( x \in z_1 \) through a sequence of \( t \) clock
ticks (for some \( t \geq 0 \)). \( x \in X_{F_i} \) because \( \tau \) transitions do not change the system condition. If the TDES does not generate any new output for all future time, then \( \text{Lim}(\{x\}) \neq \emptyset \). If, on the other hand, after \( t' \geq t_{\text{max}} + 1 \) ticks the TDES enters some \( q' \), with \( \lambda_\tau(q') \neq \lambda_\tau(q) \), then \( q' \in X \) and \( t + t' \in \mathcal{T}(x, q') \). Since \( t + t' \geq t_{\text{max}} + 1 \), \( \sup \mathcal{T}(x, q') = \infty \). Therefore, there exists a path in \( G \), from \( x \) to \( q' \) containing a cycle or a transition for which the transition-time set \( \mathcal{T}(\cdot, \cdot, \cdot) \) is unbounded. Thus, \( \text{Lim}(\{x\}) \neq \emptyset \).

\( e = 2 \). It follows from (i) that when the output \( y \) was generated, the system entered some \( x_{in} \in x \cap X_{F_i} \). By (iii), either \( \mathcal{T}(x_{in}, x) = \emptyset \) for all \( x \) such that \( \lambda_\tau(x) \neq \lambda_\tau(x_{in}) \), or there exists \( x \) such that \( \sup \mathcal{T}(x_{in}, x) = \infty \). In the first case, the system can stay indefinitely in \( R(s, \{x_{in}\}) \). Therefore, \( \text{Lim}(\{x_{in}\}) \neq \emptyset \). In the second case, there exists a path in \( G \) from \( x_{in} \) to \( x \) containing a cycle or a transition with unbounded transition-time set \( \mathcal{T}(\cdot, \cdot, \cdot) \). Therefore, \( \text{Lim}(\{x_{in}\}) \neq \emptyset \).

Therefore \( \text{Lim}(z) \cap \text{Lim}(X_{F_i}) \neq \emptyset \) since \( \text{Lim}(\{x_{in}\}) \subseteq \text{Lim}(z) \) and \( \text{Lim}(\{x_{in}\}) \subseteq \text{Lim}(X_{F_i}) \). Now, Lemma 3.11 implies \( \text{Lim}(z) \subseteq \text{Lim}(X_{F_i}) \subseteq X_{F_i} \). Also, by Lemmas 3.10 and 3.12,

\[
\kappa\left(\bigcup\{z' \mid \mathcal{T}(z, z') \neq \emptyset \& \sup \mathcal{T}(z, z') \geq t_{\text{max}} + 1\}\right) = \{F_i\}
\]

if \( \bigcup\{z' \mid \mathcal{T}(z, z') \neq \emptyset \& \sup \mathcal{T}(z, z') \geq t_{\text{max}} + 1\} \neq \emptyset \). As a result,

\[
\kappa(\text{Pred}(z, t_{\text{max}} + 1)) = \{F_i\}.
\]

Lemma 3.14 Suppose the diagnoser reaches a state \( z = (z, e) \). Assume that (i) the failure event \( F_i \) occurs after the diagnoser enters \( z \), (ii) Eq. (3.6) holds, and (iii) no new output is generated for at least \( t_{\text{max}} + 1 \) clock ticks following the failure event \( F_i \). Then

\[
\kappa(\text{Pred}(z, t_{\text{max}} + 1)) = \{F_i\}.
\]
Proof. Let $x_i$ be the state of the system immediately after the failure event. Then $x_i \in \text{Rch}(z) \cap X_{F_i}$. By (iii), either $\mathcal{T}(x_i, x) = \emptyset$ for all $x$ such that $\lambda(x) \neq \lambda(x_i)$, or there exists $x$ such that $\sup \mathcal{T}(x_i, x) = \infty$. In the first case, the system can stay indefinitely in $\text{Rch}({x_i})$. Therefore, $\text{Lim}({x_i}) \neq \emptyset$. In the second case, there exists a path in $G$ from $x_i$ to $x$ containing a cycle or a transition with unbounded transition-time set $\mathcal{T}(\cdot, \cdot, \cdot)$. Therefore, $\text{Lim}({x_i}) \neq \emptyset$. Therefore $\text{Lim}(z) \cap \text{Lim}(X_{F_i}) \neq \emptyset$, since $\text{Lim}({x_i}) \subseteq \text{Lim}(z) = \text{Lim}(\text{Rch}(z))$ and $\text{Lim}({x_i}) \subseteq \text{Lim}(X_{F_i})$. It follows from Lemma 3.11 that $\text{Lim}(z) \subseteq \text{Lim}(X_{F_i}) \subseteq X_{F_i}$ and from Lemmas 3.10 and 3.12 that

$$\kappa(\bigcup \{z' \mid \mathcal{T}(z, z') \neq \emptyset \& \sup \mathcal{T}(z, z') \geq t_{\max} + 1\}) = \{F_i\}$$

if $\bigcup \{z' \mid \mathcal{T}(z, z') \neq \emptyset \& \sup \mathcal{T}(z, z') \geq t_{\max} + 1\} \neq \emptyset$. As a result,

$$\kappa(\text{Pred}(z, t_{\max} + 1)) = \{F_i\}.$$

Proof of Theorem 3.9.

(Sufficiency) Let $z_1$ denote

(a) the state of the diagnoser after the diagnoser is initialized and the first output is read if the diagnoser is started after the failure or

(b) the state of the diagnoser immediately after the failure event if the diagnoser was started some time before the occurrence of the failure $F_i$.

From this point on, in general, the diagnoser will make the transitions $z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow \cdots$ as a result of the generation of the output sequence $y_2y_3\ldots$. Let $t_{1,2}$ denote the number of the clock ticks between the time the diagnoser enters $z_1$ to the time it makes the transition to $z_2$ in case (a) above, and the number of the clock ticks from the occurrence of failure $F_i$ to the transition to $z_2$ in case (b). Also, for $k \geq 2$, let $t_{k,k+1}$ denote the number of the clock ticks the diagnoser stays at $z_k$ before it moves to $z_{k+1}$. 

95
Let $c_i = \Sigma \{|z \cap X_{F_i}| \mid (z, 2) \in Z \& z \text{ is } F_i\text{-uncertain}\}$. It follows from condition 2 of the theorem that the diagnoser can make at most $c_i - 1$ consecutive transitions among $F_i$-uncertain states of the form $z = (z, 2)$. Also note that $z_1$ may be neither $F_i$-uncertain nor $F_i$-certain.

After the diagnoser enters $z_1$, depending on the output sequence, one of the following cases will happen.

(a) No new output is generated and the diagnoser remains in $z_1$ indefinitely. In this case, by Lemmas 3.13 and 3.14, 

$$\kappa(\text{Pred}(z_1, t_{\text{max}} + 1)) = \{F_i\};$$

i.e., the failure will be diagnosed in at most $t_{\text{max}} + 1$ clock ticks.

(b) The system generates at least $c_i + 1$ new output symbols and with the generation of each new symbol, the diagnoser makes a transition.

1. Assume $t_{k,k+1} \leq t_{\text{max}}$ ($k \geq 1$). It follows from condition 2 that for $k \geq c_i + 2$, $z_k$ is $F_i$-certain and therefore, the failure will be diagnosed in at most $(c_i + 1)t_{\text{max}}$ ticks.

2. Assume for some $k \geq 1$, $t_{k,k+1} \geq t_{\text{max}} + 1$. Let $k_0$ be the smallest such $k$. If $k_0 \geq c_i + 2$, then by an argument similar to that of Case 1, we can show that the failure will be diagnosed in at most $(c_i + 1)t_{\text{max}}$ ticks. If $k_0 \leq c_i + 1$, then by Lemmas 3.13 and 3.14, the failure will be detected and isolated in at most $(k_0 - 1)t_{\text{max}} + t_{\text{max}} + 1 \leq (c_i + 1)t_{\text{max}} + 1$ clock ticks.

(c) The system generates $n$ new output symbols, with $1 \leq n \leq c_i$ and as a result, the diagnoser makes $n$ ($1 \leq n \leq c_i$) transitions (This case happens only if $c_i \geq 1$).

1. Assume $t_{k,k+1} \leq t_{\text{max}}$ ($k \geq 1$). By Lemma 3.13, the failure will be detected and isolated in at most $nt_{\text{max}} + t_{\text{max}} + 1 \leq (c_i + 1)t_{\text{max}} + 1$ ticks ($t_{\text{max}} + 1$ ticks after the diagnoser enters $z_{n+1}$).
2. Assume for some \( k \geq 1 \), \( t_{k,k+1} \geq t_{\text{max}} + 1 \). Let \( k_0 \) be the smallest such \( k \). By Lemmas 3.13 and 3.14, the failure will be diagnosed in at most 
\[(k_0 - 1)t_{\text{max}} + t_{\text{max}} + 1 \leq c_it_{\text{max}} + 1 \text{ ticks.}\]

In summary, the failure is always diagnosed in at most \((c_i + 1)t_{\text{max}} + 1 \text{ ticks.}\)

(Necessity) Let us suppose condition 1 does not hold. Then there exist \( x \in \text{Lim}(X_{F_i}) \) and \( x' \in \left(\text{Lim}(X_N) \cap X_N\right) \cup \left(\bigcup_{i \neq j} \text{Lim}(X_{F_j})\right) \), with \( \lambda(x) = \lambda(x') \). Suppose after the occurrence of the failure \( F_i \), the system enters \( x \) and at this time the diagnoser is initialized and the output symbol \( \lambda(x) = \lambda(x') \) is read. Therefore, \( x, x' \in z_1 \). Since \( x \in \text{Lim}(X_{F_i}) \), the system can stay indefinitely in \( \text{Lim}(X_{F_i}) \) without generating any new output. But \( x, x' \in z_1 \), therefore \( x, x' \in \text{Lim}(z_1) \); hence \( \text{Pred}(z, t) \) will be \( F_i \)-uncertain for all \( t \geq 0 \). As a result, \( F_i \) cannot be diagnosed.

Let us assume condition 1 is true but condition 2 of the theorem does not hold. Using an argument similar to that given in the proof of Theorem 2.1, we can show that there exists a trajectory for the system leading to states in \( X_{F_i} \) such that the corresponding output sequence throws the diagnoser into a cycle of \( F_i \)-uncertain states and keeps it there indefinitely. Therefore \( F_i \) is not time–diagnosable.

\[ \square \]

Example 3.1 - A Simple TDES (Cont’d)

For this TDES, \( \text{Lim}(X_N) = \text{Lim}(X_F) = \emptyset \) since it never stops generating new output symbols. There are no \( F \)-uncertain, hence no \( F \)-indeterminate cycles in the diagnoser (Fig. 3.10). Therefore, \( F \) is time–diagnosable. This is expected since the timings of the output sequence in the normal and faulty modes are different.

\[ \square \]

Example 3.2 - Neutralization Process (Cont’d)

For the neutralization process (Figures 3.7 and 3.8)

\[ \text{Lim}(X_N) \cap X_N = \emptyset, \quad \text{Lim}(X_{F_1}) = \{1'\}, \quad \text{Lim}(X_{F_2}) = \{1'', 1.1''\}. \]

Obviously, Eq. (3.6) holds for \( i = 1, 2 \). Furthermore, there are no \( F_1 \) or \( F_2 \)-uncertain cycles, hence, neither \( F_1 \) nor \( F_2 \)-indeterminate cycles in the diagnoser. As a result, both \( F_1 \) and \( F_2 \) are time–diagnosable.
The reader can verify that if $V_1$ gets stuck–open, then in at most $u + u' + 2$ clock ticks, the output symbol $(C_5, L_3, a)$ will be generated and the diagnoser will produce the $F_1$-certain estimate $z = \{17'\}$. On the other hand, if $V_1$ gets stuck–closed, then in at most $3u + 3$ ticks, the diagnoser will generate $z = \{1, 1''\}$ and $\text{Pred}(z, 3) = \{1'', 1.1''\}$ will indicate the occurrence of $F_2$. Hence, $F_2$ will be diagnosed in at most $3u + 6$ clock ticks. Therefore, both $F_1$ and $F_2$ are time-diagnosable. Note that without the timing information, $F_2$ would not have been diagnosable because once the tank is emptied and $(C_1, L_0, n)$ is generated, the system stops generating new output symbols and the estimate of the system’s condition remains ambiguous. $\square$
Chapter 4

Fault Diagnosis and Consistency in Hybrid Systems

In Chapter 2, a state-based approach to fault diagnosis in discrete-event systems was developed. In many cases, the discrete-event system is a simplified model (i.e., abstraction) of a hybrid system. Typically, the design of a diagnosis system which is based on a discrete-event model is verified through simulation and test under a large, but nevertheless finite, number of operating conditions of the hybrid system. In this chapter, we attempt to develop a more systematic approach to the above verification problem. Towards this end, we examine the question of whether a (high-level) DES model contains enough information about the low-level hybrid model for the purpose of fault diagnosis; i.e., whether the models are consistent. This would mean that the analysis and design based on the high-level (DES) model will be equivalent to that performed at the low-level.

In Section 4.1, we introduce the class of hybrid systems that are of interest to us. Then in Section 4.2, we extend our framework for fault diagnosis in DES models to hybrid systems. Based on this framework, we will define the notion of ‘consistency’ between the high-level and low-level models. Section 4.3 presents a set of sufficient conditions for consistency and a semi-algorithmic method for obtaining suitable high-level models for the low-level hybrid systems.
4.1 Plant Model

Let us assume that the plant under control, i.e., plant along with continuous controllers and DES supervisors, can be modelled as a hybrid automaton $H = (X, \Sigma, \Theta, \Phi, X_0, Y, \lambda)$, with:

- $X \subseteq Q \times \mathbb{R}^n$, the reachable part of the state set
- $Q$ the finite set of discrete states or modes
- $\mathbb{R}^n$ the set of continuous-variable components of states
- $\Sigma$ the finite set of events
- $\Theta \subseteq Q \times \text{Pwr}(\mathbb{R}^n) \times \Sigma \times \{\psi : \mathbb{R}^n \to \mathbb{R}^n\} \times Q$, the set of transitions
- $\Phi = \{ f_q : \mathbb{R}^n \to \mathbb{R}^n \ | \ q \in Q\}$, the set of functions describing the dynamics of the continuous-variable component of the state
- $X_0 \subseteq Q_0 \times \Xi_0$, the set of initial conditions $(x(0) \in X_0)$, with $Q_0 \subseteq Q$ and $\Xi_0 \subseteq \mathbb{R}^n$
- $Y$ the finite set of output symbols
- $\lambda : X \to Y$, the output map.

The state of the system $x = (q, \xi)$ has two components: the mode $q \in Q$ and the continuous-variable component $\xi \in \mathbb{R}^n$. At a mode $q$, $\xi$ evolves according to $\dot{\xi} = f_q(\xi)$. At some time $t$, when the system is at the mode $q$ and the continuous-variable component of the state $\xi(t) \in \Xi$, a transition $(q, \Xi, \sigma, \psi, q')$ is enabled. If the event $\sigma$ occurs, then, upon transition, the mode of the system becomes $q'$ and the continuous-variable component $\psi(\xi(t))$. We call the function $\psi$ the assignment function of the transition.

It is assumed that the event set can be partitioned according to $\Sigma = \Sigma_f \cup \Sigma_e$. Suppose at a time $t$, a transition $(q, \Xi, \sigma, \psi, q')$ is enabled. If $\sigma \in \Sigma_f$ and no other event occurs, then $\sigma$ will occur. If $\sigma \in \Sigma_e$, then as long as $\xi \in \Xi$, $\sigma$ may occur, i.e., $\sigma$ is simply enabled.

Suppose at $t = t_0$, $x(t_0) = (q_0, \xi(t_0))$ (Fig. 4.1). The continuous-variable component will evolve on $\xi_0(t)$ which satisfies $\dot{\xi}_0 = f_{q_0}(\xi_0)$ (with $\xi_0(t_0) = \xi(t_0)$, by definition) until at some $t = t_1$, a transition occurs and the system jumps to the mode $q_1$ and
the continuous–variable component becomes $\psi(\xi_0(t_1))$. Then the continuous–variable component will follow $\dot{\xi}_1 = f_{q_1}(\xi_1)$, with $\xi_1(t_1) := \psi(\xi_0(t_1))$, until at some time $t = t_2$ another transition changes the mode to $q_2$ and the continuous–variable component becomes $\psi(\xi(t_2))$. The evolution of the mode and the continuous–variable components will continue in a similar fashion after $t = t_2$. The state of the system over the interval $[t_i, t_{i+1}]$ ($i \geq 0$) will be:

$$x_i(t) = (q_i, \xi_i(t)).$$

(4.1)

If $t_i = t_{i+1}$, then after the transition at $t = t_i$, another transition has occurred right away; i.e., two transitions have occurred back–to–back. We assume that in the hybrid system $H$, only a finite number of transitions may occur back–to–back. This is similar to the activity–loop–free assumption for timed discrete–event systems (Sec. 3.1).

We call the sequence of functions $x_i(\cdot)$ given by Eq.(4.1) a state trajectory of the system. Any other sequence of functions, defined on a sequence of closed intervals, that satisfies the dynamics of the system will also be called a state trajectory.

**Definition 4.1** Suppose we are given $t_0, t_1, t_2, \ldots, t_i, \ldots$ with $t_i \leq t_{i+1}$ for $i \geq 0$ and functions $x_i(\cdot)$ defined on the closed intervals $^1 [t_i, t_{i+1}]$ (for $i \geq 0$) with

$$x_i(t) = (q_i, \xi_i(t)) \quad (t_i \leq t \leq t_{i+1}),$$

$^1$If the sequence of functions $x_i(\cdot)$ is finite and of length, say $m$, then the last interval will be either a closed interval $[t_{m-1}, t_m]$, or an unbounded interval $[t_{m-1}, \infty)$. 

101
\[ q_i \in Q, \]
\[ \xi_i(t) \in \mathbb{R}^n \]

such that
\[ \dot{\xi}_i(t) = f_{q_i}(\xi_i(t)) \quad t_i \leq t \leq t_{i+1} \quad \forall i \geq 0. \]

Also, assume that there exist transitions from \( q_i \) to \( q_{i+1} \) \((i \geq 0)\), i.e.,
\((q_i, \Xi_i, \sigma_i, \psi_i, q_{i+1}) \in \Theta \) such that \( \xi_i(t_{i+1}) \in \Xi_i \) and \( \psi(\xi_i(t_{i+1})) = \xi_{i+1}(t_{i+1}) \). Then the sequence of functions \( x_i(\cdot) \) defined on the intervals \([t_i, t_{i+1}] \) \((i \geq 0)\) is a state trajectory of the hybrid system.

We can think of \( q, \xi \) and \( x \) as functions of time \( t \) and the index \( i \). At jump points \( t_i \), in particular, the value of \( i \) is required for determining \( x = (q, \xi) \). In the following, whenever the value of \( i \) is of no importance to the discussion, we simply write \( x(t), q(t) \) and \( \xi(t) \) to refer to the values of the state and its components, with the understanding that these are the values of the state and its components at time \( t \) in some interval.

In this work, we assume that at a mode \( q \), the output \( \lambda(x) \) is constant; i.e., \( \lambda \) is a function of \( q \) only: \( \lambda((q, \xi)) = \lambda((q, \xi')) \) for all \((q, \xi), (q, \xi') \in X \). This simply means that output changes are treated as transitions (just as in finite-state DES in Ch. 2). For example, a change of output from \( y^1 \) to \( y^2 \) is a result of some transition \((q_1, \Xi_{12}, \sigma_{12}, \psi_{12}, q_2) \) from a mode \( q_1 \) to \( q_2 \), with \( \sigma_{12} \) denoting the event corresponding to the output change. Typically, \( \psi_{12}(\cdot) \) is the identity map: \( \psi_{12}(\xi) = \xi \). \( \Xi_{12} \) is the boundary between the regions in \( \mathbb{R}^n \) where the output is \( y^1 \) and \( y^2 \). As a result of this assumption, the continuous-variable component of the state, \( \xi(t) \), is defined and known after any output change, and given by the assignment function \( \psi(\cdot) \) of the corresponding transition.

If the hybrid system does not satisfy the above assumption, it can be transformed to a new system having the same trajectories in which the assumption holds. Suppose \( H' = (X', \Sigma', \Theta', \Phi', X'_0, Y', \lambda') \) is the original system. The transformed system \( H = (X, \Sigma, \Theta, \Phi, X_0, Y, \lambda) \) can be obtained as follows. Let \( Y = Y' \). Suppose at a mode \( q \),
the system can generate $m \geq 2$ different output symbols $y^1, \ldots, y^m$ and let

$$B_i := \{ \xi \in \mathbb{R}^n \mid \lambda(q, \xi) = y^i \}, \quad 1 \leq i \leq m.$$ 

In $H$, split the mode $q$ into $m$ modes $q_1, \ldots, q_m$ in the following way:

- For any $\xi_0 \in \mathbb{R}^n$, $(q_i, \xi_0) \in X_0$ if and only if $(q, \xi_0) \in X'_0$ and $\xi_0 \in B_i$;
- For all $\xi \in \mathbb{R}^n$, $\lambda((q_i, \xi)) := y^i$;
- In the set of discrete states, replace $q$ with $q_1, \ldots, q_m$;
- For all $\xi \in \mathbb{R}^n$: $f_{q_i}(\xi) := f_q(\xi)$;
- If in $H'$, $(q, \Xi, \sigma, \psi, q') \in \Theta'$ with $q \neq q'$, then $(q_i, \Xi \cap B_i, \sigma, \psi, q') \in \Theta$ if and only if $\Xi \cap B_i \neq \emptyset$ (i.e., the transition can be enabled somewhere in $B_i$);
- If in $H'$, $(q_i, \Xi, \sigma, \psi, q) \in \Theta'$, then $(q_i, \Xi \cap \psi^{-1}(B_j), \sigma, \psi, q_j) \in \Theta$ for $i, j \in \{1, \ldots, m\}$ if and only if $\Xi \cap B_i \cap \psi^{-1}(B_j) \neq \emptyset$ (i.e., a transition from a point in $B_i$ to a point in $B_j$ is feasible);
- For $i, j \in \{1, \ldots, m\}$ with $i \neq j$, suppose the regions $B_i$ and $B_j$ have a common boundary $B_{ij} \neq \emptyset$. For simplicity, we assume that $B_{ij} \subseteq B_i$ or $B_{ij} \subseteq B_j$.

Case 1: $B_{ij} \subseteq B_i$. Let $\Xi$ be the set of points in $B_{ij}$ where state trajectories enter $B_j$, i.e.,

$$\{ \xi_0 \mid \xi_0 \in B_{ij} \& (\exists t_0 > 0 \& \xi : [0, t_0] \rightarrow \mathbb{R}^n : \xi(t) = f_q(\xi(t)) \& \xi(0) = \xi_0 \& \xi(t) \in B_j (0 < t \leq t_0)) \}.\]
Figure 4.2: Examples of boundaries: $B_{1,2} = \{(\xi_1, \xi_2 \mid \xi_1 = c \& b < \xi_2 < a)\}$ and $B_{1,2,3,4} = \{(b, c)\}$.

If $\Xi \neq \emptyset$, then we let $(q_i, \Xi, \sigma_{ij}, \text{id}, q_j) \in \Theta$, where by definition, $\sigma_{ij}$ denotes output change from $y^i$ to $y^j$ and $\sigma_{ij} \in \Sigma_f$, and $\text{id}$ denotes the identity map (i.e., $\psi(\xi) = \xi$, for all $\xi \in \mathbb{R}^n$).

Case 2: $B_{ij} \subseteq B_j$. Let $\Xi := B_{ij}$. We let $(q_i, \Xi, \sigma_{ij}, \text{id}, q_j) \in \Theta$, where by definition, $\sigma_{ij}$ denotes output change from $y^i$ to $y^j$ and $\sigma_{ij} \in \Sigma_f$.

- Let $B_{ijk_1...k_l}$ ($l \geq 1$) denote the common boundary of the regions $B_i, B_j, B_{k_1}, \ldots, B_{k_l}$ (See Fig. 4.2 for an example).

For simplicity, we assume (i) $B_{ijk_1...k_l} \subseteq B_i$, or (ii) $B_{ijk_1...k_l} \subseteq B_j$, or (iii) $B_{ijk_1...k_l} \subseteq B_r$, for some $r \in \{1, \ldots, l\}$. In the first two cases, we add transitions to $\Theta$ as in Cases 1 and 2 above for $B_{ij}$. In the third case, obviously there will be no transitions from $B_i$ to $B_j$ through $B_{ijk_1...k_l}$.

- $\Theta$ contains no transitions other than those mentioned above.

In this way, we split all of the modes at which the original system can generate more than one output symbol and expand the sets $Q', \Sigma', \Phi'$ to $Q, \Sigma, \Phi$ to accommodate the new modes, events and vector fields.

Let us consider the simple example of Fig. 4.3. Here $\sigma \in \Sigma_e' = \Sigma'$ and for the transition $q \rightarrow q'$, $\psi(\cdot) = \text{id}$, and $\Xi = \{\xi \mid \xi = 1\}$. $X_0' = \{(q, 0)\}$.

104
Case 1. Suppose \( \lambda(q, \xi) = y^1 \) if \( \xi \leq 1 \) and \( \lambda(q, \xi) = y^2 \) if \( \xi > 1 \), and let \( \lambda(q', \xi) = y^3 \).

The transformed system is given in Fig 4.4. Here \( B_1 = \{\xi \leq 1\} \) and \( B_2 = \{\xi > 1\} \). For the transition \( q_1 \rightarrow q_2 \), \( \Xi = \{\xi = 1\} \). Also \( X_0 = \{(q_1, 0)\} \).

Case 2. Suppose \( \lambda(q, \xi) = y^1 \) if \( \xi < 1 \) and \( \lambda(q, \xi) = y^2 \) if \( \xi \geq 1 \). Also \( \lambda(q', \xi) = y^3 \), as in Case 1. The transformed system is given in Fig. 4.5 where \( B_1 = \{\xi < 1\} \), \( B_2 = \{\xi \geq 1\} \). For the transition \( q_1 \rightarrow q_2 \), \( \Xi = \{\xi = 1\} \). Also \( X_0 = \{(q_1, 0)\} \).

In this case, the output must change before \( \sigma \) can occur.

The model \( H \) describes the behaviour of the system in both normal and faulty situations. Suppose there are \( p \) failure modes \( F_1, \ldots, F_p \). The event set \( \Sigma \) includes failure events. For brevity, we consider single-failure scenarios only. As with fault diagnosis in finite-state automata, our results can be easily generalized to the case of simultaneous occurrence of two (or more) failure modes.

Let \( K := \{N, F_1, \ldots, F_p\} \) denote the condition set of the system. It is assumed that the state set \( X \) can be partitioned according to the condition of the system:
Define the condition map $\kappa : X \to \mathcal{K}$ such that for every $x \in X$, $\kappa(x)$ is the condition of the system at the state $x$: $\kappa(x) = N$ if $x \in X_N$, and $\kappa(x) = F_i$ if $x \in X_{F_i}$ ($i \in \{1, \ldots, p\}$). Also extend the definition of $\kappa$ to the subsets of $X$: $\kappa(z) = \cup\{\kappa(x) \mid x \in z\}$, for any $z \subseteq X$. The condition of a hybrid system changes as a result of failure (or perhaps repair or recovery) events. Therefore, $\kappa$ is only a function of $q$ (mode), not $\xi$ (continuous-variable component).

In fault diagnosis for hybrid systems (similar to the case of finite-state automata in Ch. 2), we use the output sequence $(y_1y_2y_3 \ldots)$ to find the condition of the system. If a transition (other than output changes) is observable, then by defining new output symbols and introducing new transitions (if necessary), the information about the occurrence of the transition can be transferred and included in the output map (in the same way described in Sec. 2.1 for finite-state automata). Therefore, from now on, we shall assume that the output sequence contains all of the information available about the system.

**Example 4.1 - Water Tank**

Water is charged into a tank (Fig. 4.6) at a (volume) flow rate of $q_0$ (when the control valve is open). The tank is discharged at a rate of $q = k\sqrt{h}$, where $k$ is a constant and $h$ the water level. When water is at or below the lower threshold $L$, the controller opens the valve and when water is at or above $U$, the controller closes the valve.
Initially, the tank is empty and the valve is closed.

Let $A$ be the cross-sectional area of the tank and $h_m := (q_0/k)^2$. When the valve is open

$$
\dot{h} = \frac{q_0 - q}{A} = \frac{k}{A}(\sqrt{h_m} - \sqrt{h}),
$$

(4.2)

and when it is closed

$$
\dot{h} = -\frac{q}{A} = -\frac{k}{A}\sqrt{h}.
$$

(4.3)

It is assumed that the valve may become stuck–open (the event $F$). Let us denote the regions $\{0 \leq h \leq L\}, \{L < h < U\}$ and $\{U \leq h \leq h_m\}$ by $H_0, H_1$ and $H_2$. We assume that a sensor measures water level and generates one of the symbols $H_0$, $H_1$ and $H_2$ accordingly. Let $H_{ij}$ denote the transition from region $H_i$ to $H_j$. The state transition graph of the controller is depicted in Fig. 4.7. Here VO and VC refer to the commands “valve open” and “valve closed”, respectively. We assume that the state of the controller is available for fault diagnosis. Therefore, for this system
Figure 4.8: Example 4.1. Hybrid automaton.

<table>
<thead>
<tr>
<th>$q$</th>
<th>Output</th>
<th>$q$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$(H_0, C1)$</td>
<td>$q_7$</td>
<td>$(H_0, C1)$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$(H_0, C2)$</td>
<td>$q_8$</td>
<td>$(H_0, C2)$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(H_1, C2)$</td>
<td>$q_9$</td>
<td>$(H_1, C2)$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$(H_2, C3)$</td>
<td>$q_{10}$</td>
<td>$(H_2, C3)$</td>
</tr>
<tr>
<td>$q_5$</td>
<td>$(H_2, C4)$</td>
<td>$q_{11}$</td>
<td>$(H_2, C4)$</td>
</tr>
<tr>
<td>$q_6$</td>
<td>$(H_1, C4)$</td>
<td>$q_{12}$</td>
<td>$(H_1, C4)$</td>
</tr>
</tbody>
</table>

Table 4.1: Example 4.1. Output map.

$Y \subseteq \{H_0, H_1, H_2\} \times \{C_1, C_2, C_3, C_4\}$.

The hybrid automaton for the water tank is shown in Fig. 4.8. In this system, $\Sigma_f = \{\text{VO, VC}\}$, $\Sigma_e = \{F\}$, and the assignment functions of the transitions ($\psi(\cdot)$) are all identity maps. Following the algorithm mentioned before this example, we transform the system in Fig. 4.8 to that depicted in Fig. 4.9, in order to model output changes as transitions. In these figures, $\uparrow$ and $\psi$ refer to the equations (4.2) and (4.3), respectively. In this system, when the valve is in the normal condition and the water level increases ($\uparrow$), the output changes from $(H_0, C2)$ to $(H_1, C2)$, and from $(H_1, C2)$ to $(H_2, C3)$. As a result, the mode $\uparrow$ of the normal condition in Fig. 4.8 has been split into three modes $q_2$, $q_3$ and $q_4$ in Fig. 4.9. The rest of the modes of the system in Fig. 4.8 are split similarly. The output map is given in Table 4.1. In this system, $\xi = h$ and $Q = \{q_1, \ldots, q_{12}\}$. The assignment functions $\psi(\cdot)$ are all identity maps.

\[\square\]
4.2 Diagnoser

In our framework, the diagnoser for the hybrid system $H$ is a system (not necessarily finite-state) that takes the output sequence of the system $(y_1, y_2, \ldots, y_k)$ as input and generates at its output an estimate of the condition of the system at the time that $y_k$ was generated. Specifically, based on the output sequence up to $y_k$, a set $z_k \in 2^{Q \times \mathbb{R}^n} - \{\emptyset\}$ is calculated to which $x$ must belong at the time that $y_k$ was generated. $\kappa(z_k)$ will be the estimate of the system's condition. Upon observing $y_{k+1}$, $z_k$ will be updated to $z_{k+1}$.

**Definition 4.2** For any two states $x, x' \in Q \times \mathbb{R}^n$, we say $x'$ is **output-adjacent** to $x$ and write $x \to x'$ if $\lambda(x) \neq \lambda(x')$ and there exists a trajectory of finite duration, satisfying the hybrid system's dynamics, starting at $x$ and ending at $x'$ such that for all $x'' \neq x'$ belonging to the trajectory: $\lambda(x'') = \lambda(x)$. □

We define the diagnoser to be a machine (not necessarily finite-state) $D = (Z \cup \{z_0\}, Y, \zeta, z_0, \hat{K}, \kappa)$, where $Z \cup \{z_0\}$, $Y$ and $\hat{K} \subseteq 2^K - \{\emptyset\}$ are the state, event and
output sets of $D$. $z_0 := (z_0, 0)$ is the initial state, with $z_0 \in 2^X - \emptyset$. $Z \subseteq 2^X - \emptyset$ and is not necessarily finite. $\zeta : Z \cup \{z_0\} \times Y \to Z$ is the transition (partial) function and $\kappa : Z \cup \{z_0\} \to \hat{\mathcal{K}}$ the output map, with the definition of the condition map, $\kappa$, extended according to: $\kappa(z_0) := \kappa(z_0)$. Each diagnoser state $z$, except $z_0$, is identified with a nonempty subset of $X$. $z_0$, however, is considered to be different from other states of the diagnoser because the update law $\zeta$ at $z_0$ is different from that at other states. To distinguish $z_0$ from other states, it has been identified with a pair $(z_0, 0)$ (rather than a subset).

The diagnoser state transition $z_{k+1} = \zeta(z_k, y_{k+1})$ is given by:

\[
\begin{align*}
  z_1 &= z_0 \cap \lambda^{-1}(\{y_1\}) \\
  z_{k+1} &= \{x \mid \lambda(x) = y_{k+1} \& (\exists \ x' \in z_k : \ x' \Rightarrow x)\} \ \forall k \geq 1.
\end{align*}
\]

Therefore $z_{k+1}$ will be the set of states, having output $y_{k+1}$, that are reachable from the states in $z_k$ using trajectories along which the output is $y_k$.

As assumed in the previous section, in the model $H$, changes in the output are represented by transitions. The state of the system immediately after the output change is given by the corresponding transition; i.e., if $(q, \Xi, \sigma, \psi, q')$ is the transition, then for any $x = (q, \xi)$ with $\xi \in \Xi$, $(q', \psi(\xi))$ will be the state of the system immediately after the transition (output change).

$z_0$, the initial state estimate, contains the information available about the state of the system at the time that the diagnoser is initialized, before the reading of sensors begins. Similar to the case of finite–state automata, usually $z_0 = X$.

The construction of the diagnoser for hybrid automata is similar to that for finite–state automata (Sec. 2.2) except that for hybrid automata, the state estimates ($z_k$) are not necessarily finite sets. For updating the state of the diagnoser, $z_k$, information about output–adjacent states is required (refer to the definition of the update law $\zeta$). Similar to the case of finite–state automata, this information can be gathered in a reachability transition system (RTS). The RTS corresponding to the hybrid transition system $H$ is defined to be $\hat{H} = (X, R, Y, \lambda)$, where the state set $X$, output
set $Y$ and output map $\lambda$ are the same as those of the original system, and the transition relation $R \subseteq X \times X$ is given by

$$R = \{ (x_1, x_2) \mid x_1, x_2 \in X, \ x_1 \Rightarrow x_2 \};$$

i.e., $(x_1, x_2) \in R$ if and only if $x_2$ is output-adjacent to $x_1$. Note that $\tilde{H}$ is not a finite transition system since $X$ and (in general) $R$ are not finite sets.

**Example 4.1 - Water Tank (Cont'd)**

The RTS and the diagnoser for the water tank system are given in Table 4.2 and Fig. 4.10. In this example, the diagnoser has a finite number of states. The operation of the diagnoser is similar to that of the diagnosers of Ch. 2. 

\[ \square \]
Table 4.2: Example 4.1. Reachability transition system.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$\xi$</th>
<th>$\lambda((q, \xi))$</th>
<th>Output–adjacent state(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$\in {0, L}$</td>
<td>$(H_0, C1)$</td>
<td>$(q_2, \xi), (q_8, \xi)$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$\in [0, L]$</td>
<td>$(H_0, C2)$</td>
<td>$(q_3, L), (q_9, L)$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\in [L, U]$</td>
<td>$(H_1, C2)$</td>
<td>$(q_4, U), (q_{10}, U)$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$U$</td>
<td>$(H_1, C2)$</td>
<td>$(q_4, U)$</td>
</tr>
<tr>
<td>$q_5$</td>
<td>$U$</td>
<td>$(H_2, C3)$</td>
<td>$(q_5, U), (q_{11}, U)$</td>
</tr>
<tr>
<td>$q_6$</td>
<td>$U$</td>
<td>$(H_2, C4)$</td>
<td>$(q_6, U)$</td>
</tr>
<tr>
<td>$q_7$</td>
<td>$\in (L, U]$</td>
<td>$(H_1, C4)$</td>
<td>$(q_1, L), (q_{11}, U)$</td>
</tr>
<tr>
<td>$q_8$</td>
<td>$L$</td>
<td>$(H_1, C4)$</td>
<td>$(q_1, L)$</td>
</tr>
<tr>
<td>$q_9$</td>
<td>$\in [0, L]$</td>
<td>$(H_0, C1)$</td>
<td>$(q_8, \xi)$</td>
</tr>
<tr>
<td>$q_{10}$</td>
<td>$\in [L, U]$</td>
<td>$(H_1, C2)$</td>
<td>$(q_{10}, U)$</td>
</tr>
<tr>
<td>$q_{11}$</td>
<td>$\in [U, h_m]$</td>
<td>$(H_2, C4)$</td>
<td>—</td>
</tr>
<tr>
<td>$q_{12}$</td>
<td>$\in [L, U]$</td>
<td>$(H_1, C4)$</td>
<td>$(q_{11}, U)$</td>
</tr>
</tbody>
</table>

4.3 Diagnosability and Consistency

Let us assume that the failure modes are permanent. The definitions of fault–certain and fault–uncertain states of the diagnoser are similar to those of the diagnoser for finite–state automata (Sec. 2.3). In a single–failure scenario, $z$ is $F_i$–certain if and only if $\kappa(z) = \{F_i\}$, and $F_i$–uncertain if and only if $F_i \in \kappa(z)$ but $\kappa(z) \neq \{F_i\}$. The definition of diagnosability in hybrid systems is also similar to that given in Sec. 2.3 except that “bounded number of events” is replaced with “finite time”.

**Definition 4.3** A permanent failure mode $F_i$ is **diagnosable** if following both the occurrence of the failure and initialization of the diagnoser, $F_i$ can be detected and isolated (i.e., the diagnoser reaches an $F_i$–certain state) in finite time.

In the above definition, no assumption is made about the system’s condition at the time that the diagnoser was initialized; i.e., the diagnoser could have been initialized either before or after the occurrence of the failure. In the water tank system (Example 4.1), the failure (valve stuck–open) can be detected only if it occurs when $L < h < U$ and the valve is closed. In this case, the output will be $(H_1, C4)$, followed by $(H_2, C4)$.
in less than \((U - L)/\dot{h}_{\text{min}}\) units of time where \(\dot{h}_{\text{min}} = k(\sqrt{h_{\text{rm}}^2} - \sqrt{U})/A\). Upon the generation of \((H_2, C4)\), the diagnoser enters the \(F\)-certain state \(z = \{(q11, U)\} \).

**Remark 4.1** Let \(t_{\text{init}}\) and \(t_{F_i}\) denote the time when the diagnoser is initialized and the time when the failure \(F_i\) occurs, respectively. Let \(t_0 = \max\{t_{\text{init}}, t_{F_i}\}\). We refer to the time (from \(t_0\)) that it takes the diagnoser to detect and isolate a failure mode as the **diagnosis time**, and denote it by \(T_d\). We define \(T_d = \infty\) when the failure is not diagnosed. In general, \(T_d\) depends on \(t_{\text{init}}\) and the trajectory the systems evolves on (Here, the initial state estimate, \(z_0\), is considered a fixed parameter of the diagnoser).

According to Def. 4.3, \(F_i\) is diagnosable if \(T_d\) is a finite number for any \(t_{\text{init}}\) and any state trajectory that either enters \(X_{F_i}\) or is entirely in \(X_{F_i}\). Later in the following section, we discuss the cases where we want \(T_d\) to be bounded; i.e., the cases where there exists \(T \geq 0\) such that for any \(t_{\text{init}}\) and any state trajectory (entering \(X_{F_i}\) or entirely in \(X_{F_i}\)), we have \(T_d \leq T\). \(\Box\)

So far, we have introduced the hybrid system, the corresponding diagnoser and the concept of diagnosability. The diagnoser is used for two purposes:

(i) To determine the condition of the system based on the output sequence; and

(ii) To examine the diagnosability of the failure modes.

In practice, if possible, instead of the detailed hybrid model, a simpler finite-state automaton may be used for modelling the system. In this case, using the theory developed in Chapter 2, a (high-level) finite-state diagnoser can be designed to provide answers to the above-mentioned questions (Fig. 4.11).

The high-level model (finite-state automaton) is simpler than the low-level hybrid model and contains less information about the system. At this point, we are not concerned with how the high-level model is obtained.

**Definition 4.4** The low-level and high-level models of the system as described above are said to be **consistent** (for the purpose of fault diagnosis) if and only if
- the low-level and high-level diagnosers are equivalent; i.e., they generate the same condition estimate in response to a given output sequence of the system, and

- a failure mode is diagnosable with respect to the low-level model (Def. 4.3) if and only if it is diagnosable with respect to the high-level model (Def. 2.4).

When the models are consistent, the diagnosers provide identical answers to questions (i) and (ii) above. Note that in the above definition, it is assumed that the output set of the high-level model is the same as that of the low-level model, namely, \( Y \). In some cases, \( Y \) is not given initially and a suitable output set for the systems has to be found. We do not address this issue in our work.

If the high-level model has been derived from the low-level model using a model reduction scheme that guarantees consistency, then in the process of model reduction, the information required for fault diagnosis is preserved. In the following section, we will study this matter in more detail.

### 4.4 Consistent Model Reduction

In this section, we introduce a particular set of partitions of the state set \( X \) of the hybrid system \( H \). If this set is nonempty, then we show how a consistent high-level
finite-state DES model for the hybrid system can be constructed using any of these partitions.

**Definition 4.5** Let \( \pi = \{ B^1, \ldots, B^p \} \) be a finite partition of \( X \), with \( B^i \) denoting its blocks. Also let \( P : X \rightarrow X/\pi = \bar{X} \) be the canonical projection that maps every state \( x \) to its coset in \( X/\pi \). Suppose \( x(\cdot) \) is a trajectory consisting of a sequence of functions \( x_i(\cdot) \). We call the sequence of functions \( P x_i(\cdot) \) the projection of the trajectory (on \( X/\pi \)).

In this work, for simplicity, we do not distinguish between a coset \( \bar{x} \in \bar{X} \) and the corresponding block \( P^{-1}\bar{x} \). Therefore the projection of a trajectory is simply the set of blocks that are crossed or passed through by the trajectory.

Throughout this section, we will assume that

- Assumption (a): The trajectories of the hybrid system and the partitions, including \( \ker \lambda \) and \( \ker \kappa \), satisfy the condition that the projection of any trajectory \( x(\cdot) \), is a sequence of the blocks of the partition. \(^4\)

Let \( \pi = \{ B^1, B^2, \ldots, B^{\pi_1} \} \) be a finite partition of the state set \( X \) of the hybrid system \( H \) with the following properties:

(I) \( \pi \leq \ker \kappa \wedge \ker \lambda; \)

(II) let \( B, B' \in \pi \), with \( B \neq B' \). If for some \( x_1 \in B \), there exists \( x'_1 \in B' \) and a trajectory of finite duration that starts at \( x_1 \) and ends at \( x'_1 \) and lies in \( B \cup B' \), then for any \( x_2 \in B \), there exists \( x'_2 \in B' \) and a trajectory of finite duration from \( x_2 \) to \( x'_2 \) lying in \( B \cup B' \).

Let \( \Pi^* \) be the set of partitions with the above properties. Timed automata [1] are examples of systems for which \( \Pi^* \neq \emptyset \). In general, \( \Pi^* \) might be empty however. Consider the example in Fig. 4.12. In this system, the transition \( q_1 \rightarrow q_2 \) becomes

---

\(^4\)Here is an example of a partition and a trajectory that do not satisfy the condition. Let \( \pi = \{ B^0, B^1 \} \) with \( B^0 = \text{rational numbers} \) and \( B^1 = \mathbb{R} - B^0 \). Also let \( x(t) = t \), with \( t \in (-\infty, \infty) \). The function \( y(t) \), with \( y(t) = 1 \) if \( y \in B^0 \), \( y(t) = 0 \) if \( y \in B^1 \), has an uncountable, infinite number of discontinuities. In fact, it is discontinuous everywhere in \( \mathbb{R} \).
enabled when $\xi \geq 1$ and the transition $q_2 \rightarrow q_1$ is enabled for all $\xi$. Suppose $\ker \lambda = \{(q_1) \times [0, 1), (q_1) \times [1, \infty), (q_2) \times [1, \infty)\}$ (Note that $(q_2) \times [0, 1)$ is not reachable. Any partition $\pi \in \Pi^*$ has the form $\{(q_1) \times A_1, \ldots, (q_1) \times A_{n_1}, (q_2) \times A'_1, \ldots, (q_2) \times A'_{n_2}\}$, with $A_i \subseteq [0, \infty)$ and $A'_j \subseteq [1, \infty)$ ($1 \leq i \leq n_1, 1 \leq j \leq n_2$) and $\bigcup_{i=1}^{n_1} A_i = [0, \infty)$ and $\bigcup_{j=1}^{n_2} A'_j = [1, \infty)$. Since the assignment function for $q_2 \rightarrow q_1$ is the identity map, $\pi$ satisfies (II) only if $A_i \cap A'_j \neq \emptyset \text{ or } A'_j \subseteq A_i$, for any $i$ and $j$, with $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$. This means that the partition of $[1, \infty)$ in the mode $q_2$ is finer than that in the mode $q_1$; i.e., $\{A'_j | 1 \leq j \leq n_2\} \subseteq \{A_i | A_i \subseteq [1, \infty), 1 \leq i \leq n_1\}$. Let us suppose that for practical purposes we are only interested in partitions in which the $A_i$ (resp. the $A'_j$) are finite unions of subintervals of $[0, \infty)$ (resp. $[1, \infty)$).

Since $[0, \infty)$ is unbounded, one of the $A_i$, say $A_{i_0}$, must be unbounded and contain an interval of the form $I = [a, \infty)$ (or $I = (a, \infty)$), with $a > 1$. Let $J = \{j | A'_j \subseteq I\}.$ Therefore, $I = \bigcup_{j \in J} A'_j.$ The interval $I$ under the assignment $\psi(\xi) = \frac{1}{2} \xi$ is mapped to $[a/2, \infty)$ (or $(a/2, \infty)$ if $I = (a, \infty)$). The subset $(2a, \infty)$ is mapped to $I = \bigcup_{j \in J} A'_j.$ If $I = [a, \infty)$, then we can find a sufficiently small open set $(2a - \epsilon, 2a)$ which is mapped to points outside $A_{i_0}$. If $I = (a, \infty)$, then the point $2a$ is mapped to $a \notin A_{i_0}$. Therefore, part of $(q_1) \times A_{i_0}$ is mapped to $(q_2) \times (\bigcup_{j \in J} A'_j) \subseteq (q_2) \times A_{i_0}$ and part of it to $(q_2) \times ([0, \infty) - A_{i_0}).$ This violates (II). Hence, restricting the $A_i$ and the $A'_j$ to finite unions of subintervals of $[0, \infty)$ and $[1, \infty)$, respectively, $\Pi^* = \emptyset$. Note that the infinite partition $\{(q_1) \times [0, 1), (q_1) \times [1, 2), (q_1) \times [2, 4), \ldots \} \cup \{(q_2) \times [1, 2), (q_2) \times [2, 4), \ldots \}$ satisfies properties (I) and (II).

Suppose $\Pi^* \neq \emptyset$ and $\pi \in \Pi^*$. Let $P : X \rightarrow \overline{X} = X/\pi$ be the canonical projection. For every $x \in X$, let $[x] := Px$ denote the coset corresponding to $x$. For simplicity,
instead of $x \in P^{-1} \overline{x}_1$ (for $\overline{x}_1 \in \overline{X}$), we write $x \in \overline{x}_1$. 

**Definition 4.6** Let $\pi \in \Pi^*$ and $P : X \rightarrow \overline{X} = X/\pi$ be the canonical projection. The high-level DES corresponding to $H$ is the finite-state Moore automaton $\overline{G} = (\overline{X}, \Sigma, \overline{\delta}, \overline{X}_0, Y, \lambda)$, where $\overline{X}$, $\Sigma$ and $Y$ are the state, event and output sets. $\overline{X} := X/\pi$ and the event set is a singleton $\overline{\Sigma} = \{\overline{\sigma}\}$, for some $\overline{\sigma}$. $\overline{X}_0 := PX_0$ is the set of initial states. $\overline{\delta} : \overline{X} \times \overline{\Sigma} \rightarrow 2^{\overline{X}}$ is the transition function and $\overline{\lambda} : \overline{X} \rightarrow Y$ the output map. They are defined as follows. For all $\overline{x}_1, \overline{x}_2 \in \overline{X}$, by definition $\overline{x}_2 \in \overline{\delta}(\overline{x}_1, \overline{\sigma})$ if and only if there exist $x_1$ and $x_2$ with $x_1 \in \overline{x}_1$ and $x_2 \in \overline{x}_2$ and a trajectory of finite duration from $x_1$ to $x_2$ lying in $\overline{x}_1 \cup \overline{x}_2$. For all $\overline{x} \in \overline{X}$, $\overline{\lambda}(\overline{x}) := \lambda(x)$ for any $x \in \overline{x}$. Similarly, we define $\overline{\kappa} : \overline{X} \rightarrow \mathcal{K}$ according to $\overline{\kappa}(\overline{x}) = \kappa(x)$ for any $x \in \overline{x}$. 

Since $\pi \leq \ker \lambda \wedge \ker \kappa$, $\overline{\lambda}$ and $\overline{\kappa}$ are well-defined. It was assumed in Sec. 4.1 that the output sequence contains all of the information available about the system through observation. This information is retained at the high level by the output map $\overline{\lambda}$. Therefore we do not need to differentiate between events at the high level (or the low level for that matter) and hence, let the event set of the high-level model be a singleton.

**Lemma 4.1** Suppose $x \Rightarrow x'$, for some $x, x' \in X$. Then $[x] \Rightarrow [x']$; i.e., in the high-level DES, $[x]$ and $[x']$ are output-adjacent.

**Proof.** Since $x'$ is output-adjacent to $x$, there exists a trajectory from $x$ to $x'$ on which the output is $\lambda(x)$, except at $x'$. By assumption (a), the projection of this trajectory is a sequence of states from $[x]$ to $[x']$. Therefore, in the automaton $\overline{G}$, $[x']$ is reachable from $[x]$. Let $\overline{X}_s$ be the set of the high-level states which $\overline{G}$ passes in going from $[x]$ to $[x']$. $\overline{X}_s$ is finite since $\overline{X}_s \subseteq \overline{X}$. Therefore there exists a path: $[x] = \overline{x}_1 \rightarrow \overline{x}_2 \rightarrow \ldots \rightarrow \overline{x}_m = [x']$, with $\overline{x}_i \in \overline{X}_s$ and $2 \leq m \leq |\overline{X}_s|$. Since the output of each of the states in $\overline{X}_s$, except $[x']$, is $\lambda([x])$, $\lambda(\overline{x}_i) = \lambda([x])$ $(1 \leq i \leq m - 1)$. Therefore, $[x] \Rightarrow [x']$. 

**Lemma 4.2** Suppose $\overline{x} \Rightarrow \overline{x}'$, for some $\overline{x}, \overline{x}' \in \overline{X}$. Then for any $x \in \overline{x}$, there exists $x' \in \overline{x}'$ such that $x \Rightarrow x'$. 

117
Proof. Since $\overline{x}$ and $\overline{x'}$ are output-adjacent, there exists a path $\overline{x} = \overline{x^1} \rightarrow \overline{x^2} \rightarrow \overline{x^3} \rightarrow \cdots \rightarrow \overline{x^m} = \overline{x'}$, with $m \leq |X|$ such that $\overline{\lambda(\overline{x^i})} = \lambda(\overline{x^i}) \neq \lambda(\overline{x'})$, for $1 \leq i \leq m - 1$. By property (II), there exists a trajectory of finite duration which starts at $x$, lies in $\overline{x^1} \cup \overline{x^2}$ and ends at some $x^2 \in \overline{x^2}$. Similarly, we can find a trajectory of finite duration from $x^2$ to some $x^3 \in \overline{x^3}$, lying entirely in $\overline{x^2} \cup \overline{x^3}$ and so on. In this way, a trajectory of finite duration can be constructed from $x_2$ to some $x^m \in \overline{x^m} = \overline{x'}$. On the trajectory from $x$ to $x^m$, let $x'$ be the state immediately after the output changes from $\lambda(\overline{x}^{m-1}) = \lambda(\overline{x})$ to $\lambda(\overline{x'})$ (i.e., following the transition to $\overline{x'}$). We can see that $x \Rightarrow x'$.

We define the canonical projection of a subset $z \subseteq X$ to be $Pz := \bigcup\{[x] \mid x \in z\}$. Let $\overline{D}$ be the (high-level) diagnoser designed for the high-level DES $\overline{G}$ (following the procedure given in Ch. 2), with $\overline{z}_0 := Pz_0$ as its initial state estimate ($z_0$ is the initial state estimate in the (low-level) diagnoser for the hybrid system).

**Definition 4.7** The low-level and high-level diagnosers are equivalent if for any output sequence, they produce identical sequences of estimates for the system’s condition.

**Theorem 4.3** The low-level diagnoser $D$ and the high-level diagnoser $\overline{D}$ are equivalent.

**Proof.** We have to show that for every output sequence $(y_1, y_2, \ldots, y_j)$ and any initial state estimate $z_0$, the diagnosers produce the same sequence of estimates for the condition of the system, i.e., $\kappa(z_k) = \overline{\kappa}(\overline{z}_k)$ for $0 \leq k \leq j$ ($\overline{z}_k$ is the state of $\overline{D}$). Let $\overline{\zeta}$ denote the transition (partial) function of $\overline{D}$. First we show that $Pz_k = \overline{z}_k$ (Fig. 4.13).

By definition $Pz_0 = \overline{z}_0$.

$$
Pz_1 = P(z_0 \cap \lambda^{-1} \{y_1\})
= Pz_0 \cap P(\lambda^{-1}(\{y_1\}))
= z_0 \cap \overline{\lambda}^{-1}(\{y_1\}) \quad \text{(because } \pi \leq \ker \lambda)
= \overline{z}_1.
$$
Now we prove that for $k \geq 1$ if $Pz_k = \bar{z}_k$, then $Pz_{k+1} = \bar{z}_{k+1}$.

Let $x_{k+1}$ be an element of $z_{k+1}$, i.e., $x_{k+1} \in z_{k+1}$. Then there exists $x_k \in z_k$ such that $x_k \Rightarrow x_{k+1}$. Hence by Lemma 4.1, $[x_k] \Rightarrow [x_{k+1}]$. From $Pz_k = \bar{z}_k$ it follows that $[x_k] \in \bar{z}_k$. Also $\bar{\lambda}([x_{k+1}]) = \lambda(x_{k+1}) = y_{k+1}$. As a result, $[x_{k+1}] \in \bar{z}_{k+1}$. This shows that $Pz_{k+1} \subseteq \bar{z}_{k+1}$.

Now let $\bar{x}_{k+1}$ be an element of $\bar{z}_{k+1}$, i.e., $\bar{x}_{k+1} \in \bar{z}_{k+1}$. Therefore $\bar{x}_k \Rightarrow \bar{x}_{k+1}$ for some $\bar{x}_k \in \bar{z}_k = Pz_k$. Let $x'_{k+1} \in \bar{x}_k \cap z_k \neq \emptyset$. Hence by Lemma 4.2, there exists $x'_{k+1} \in \bar{x}_{k+1}$ such that $x'_{k} \Rightarrow x'_{k+1}$. Also $\lambda(x'_{k+1}) = \bar{\lambda}(\bar{x}_{k+1}) = y_{k+1}$. Thus $x'_{k+1} \in z_{k+1}$ and as a result, $\bar{x}_{k+1} = Px'_{k+1} \in Pz_{k+1}$. This proves $z_{k+1} \subseteq Pz_{k+1}$.

Therefore, by induction, $Pz_k = \bar{z}_k$ for $0 \leq k \leq j$. Finally, $\kappa(\bar{z}_k) = \kappa(Pz_k) = \kappa(z_k)$ since $\pi \leq \ker \kappa$. \qed

Note the similarity between the above proof and the proof of the equivalence of high-level and low-level diagnosers in Ch. 2.

Next we show the equivalence of high-level and low-level diagnosabilities (Definitions 2.4 and 4.3). First we make two assumptions the validity of which depends on the partition $\pi$ and the dynamics of the hybrid system. We will use them to establish the equivalence of the two notions of diagnosability. Later, we will examine the assumptions in more detail.

- **Assumption (A):** The projection of any low-level trajectory of finite duration on the high-level state space is a finite sequence of states.

- **Assumption (B):** Suppose that the projection of a low-level trajectory on the high-level state space is a single state, say $\bar{x}$, with $\bar{\delta}(\bar{x}, \sigma) \neq \emptyset$. Then the trajectory must be of finite duration.
Assumption (A) states that over a finite interval of time, only a finite number of events can occur at the high-level DES. Note that assumption (A) is stronger than assumption (a), previously introduced in this section.

**Theorem 4.4** A failure mode is diagnosable with respect to the low-level (hybrid) model if and only if it is diagnosable with respect to the high-level (DES) model.

**Proof.** Let \( t_{F_i} \) be the instant at which the failure \( F_i \) occurs and \( t_{\text{init}} \) the instant at which the diagnosers are initialized. Let \( t_0 = \max(t_{F_i}, t_{\text{init}}) \).

(If) Suppose a failure mode \( F_i \) is not diagnosable with respect to the low-level model but diagnosable with respect to the high-level model. Then there exists a trajectory for \( t \geq t_0 \) starting at some \( x(t_0) \in X_{F_i} \) such that if the system evolves on it, the low-level diagnoser will never reach an \( F_i \)-certain state.

- If the projection of the trajectory on the high level is an infinite sequence of states, then since \( F_i \) is diagnosable with respect to the high level, the high-level diagnoser will enter an \( F_i \)-certain state after a bounded number of events are generated. Since the high-level and low-level diagnosers are equivalent (\( \kappa(z_k) = \bar{\kappa}(\bar{z}_k) \)), this is not possible.

- If the projection of the trajectory on the high level is a finite sequence of events, then by assumption (B), the last state in this sequence is a state from which transition to no other state is possible. But again since \( F_i \) is diagnosable with respect to the high level, after the time the high-level system enters the final state of the above sequence, the high-level diagnoser must be in an \( F_i \)-certain state. Since the high-level and low-level diagnosers are equivalent (\( \kappa(z_k) = \bar{\kappa}(\bar{z}_k) \)), the low-level diagnoser must also enter an \( F_i \)-certain state. By assumption (B), it takes a finite time from \( t = t_0 \) for \( \bar{x} \) to get to the last state of its path. This contradicts the assumption that \( F_i \) is not diagnosable with respect to the low-level model.

(Only if) Suppose a failure mode \( F_i \) is not diagnosable with respect to the high-level system. Then either (i) there exists a finite sequence of states \( \{\bar{x}_i\}_{0 \leq i \leq m} \) ending
at a state in $X_{F_i}$, from which there are no transitions to other states, and the resulting output sequence does not take the high-level diagnoser into an $F_i$-certain state, or (ii) there exists an infinite sequence of states $\{\overline{x}_i\}_{i \geq 0}$ and a $k \geq 0$ with $\overline{x}_i \in X_{F_i}$ for $i \geq k$ such that the resulting output sequence $\{\lambda(\overline{x}_i)\}_{i \geq 0}$ does not take the high-level diagnoser into an $F_i$-certain state. We examine the two cases separately.

(i) In this case, one can construct a trajectory of finite duration, say for $0 \leq t \leq t_f$, whose projection on the high level will be $\{\overline{x}_i\}_{0 \leq i \leq m}$. No matter how this trajectory is continued after $t \geq t_f$, no new output symbol will be generated because the projection of the rest of the trajectory will be $\overline{x}_m$. Since $F_i$ is diagnosable (with respect to the low-level model), therefore the finite duration trajectory should take the low-level diagnoser into an $F_i$-certain state; hence, by the equivalence of low-level and high-level diagnosers, the output sequence $\{\lambda(\overline{x}_i)\}_{0 \leq i \leq m}$ should take the high-level diagnoser into an $F_i$-certain state which is a contradiction.

(ii) We can construct a trajectory for $t \geq 0$ at the low level whose projection on the high level will be $\{\overline{x}_i\}_{i \geq 0}$. By assumption (A), this trajectory will not be of finite duration. Since $F_i$ is diagnosable with respect to the low-level system, there exists $t_f \geq t_0$ such that the output sequence generated between $t_{\text{init}} \leq t \leq t_f$ will take the low-level diagnoser into an $F_i$-certain state. This output sequence also takes the high-level diagnoser into an $F_i$-certain state. By assumption (A), over $[t_{\text{init}}, t_f]$ only a finite number of events can occur. But this is a contradiction.

\[\square\]

**Remark 4.2** Loosely speaking, assumptions (A) and (B) mean: "finite number of events is equivalent to finite duration". This should not come as a surprise because at the low level a failure is diagnosable if it can be detected and isolated in a "finite time" while at the high level, failure detection and isolation is expected to happen in a "finite number of events". If the two definitions are to become equivalent, then we can expect some sort of equivalence between the "real time" and "logical time". \[\square\]
Remark 4.3 Consider a state \( \bar{x} \) of a finite-state automaton for which \( \delta(\bar{x}, \bar{s}) \neq \emptyset \) for all \( \bar{s} \in \overline{S} \subseteq \overline{S} \), with \( \overline{S} \neq \emptyset \). In fault diagnosis for finite-state automata, it is implicitly assumed that if the state of the automaton reaches \( \bar{x} \), then at some point in the future one of the events in \( \overline{S} \) will occur (and therefore the information about this transition and the possible future transitions can be used for diagnosis). This is exactly what assumption (B) implies.

\[ \square \]

Remark 4.4 Diagnosability depends on system’s behaviour after the occurrence of failure. Therefore, for establishing the equivalence of high-level and low-level diagnosabilities, assumptions (A) and (B) are only required to be valid for the trajectories in \( X - X_N \) and sequences in \( X - X_N \).

\[ \square \]

Remark 4.5 In Def. 4.3, by replacing “finite time” with “bounded finite time”, we arrive at a stronger notion of (low-level) diagnosability (Please refer to Remark 4.1). If in assumption (B), “finite duration” is replaced with “bounded finite duration”, then we can show that Theorem 4.4 will still hold. The “only if” part follows from the “only if” part of Theorem 4.4 (Note that in the proof of the “only if” part of Theorem 4.4, assumption (B) is not used.). Here we prove the (if) part.

Let \( t_{F_i} \) be the instant at which the failure \( F_i \) occurs, \( t_{\text{init}} \) the instant at which the diagnosers are initialized and \( t_0 = \max(t_{F_i}, t_{\text{init}}) \). Let us assume that a failure mode \( F_i \) is not diagnosable with respect to the low-level model.

Suppose there exists a trajectory for \( t \geq t_0 \) starting at some \( x(t_0) \in X_{F_i} \) such that if the system evolves on it, the low-level diagnoser never reaches an \( F_i \)-certain state.

- If the projection of the trajectory on the high-level is an infinite sequence of states, then since \( F_i \) is diagnosable with respect to the high level, the high-level diagnoser will enter an \( F_i \)-certain state after a bounded number of events are generated. Assumption (B) implies that these bounded number of events have to occur in a bounded time. Since the high-level and low-level diagnosers are equivalent \( (\kappa(z_k) = \overline{\kappa}(\bar{z}_k)) \), this is not possible.

- If the projection of the trajectory on the high level is a finite sequence of events, then by assumption (B), the last state in this sequence is a state from which
transition to no other state is possible. But again since $F_i$ is diagnosable with respect to the high level, after the time the high-level system enters the final state of the above sequence, the high-level diagnoser must be in an $F_i$-certain state. In fact, the high-level diagnoser reaches an $F_i$-certain state before a bounded number of transitions in the high-level model. Once again by assumption (B), these transitions happen in a bounded time. Since the high-level and low-level diagnosers are equivalent ($\kappa(z_k) = \kappa(\bar{z}_k)$), the low-level diagnoser must also enter an $F_i$-certain state in a bounded time (from $t = t_0$). This contradicts the assumption that $F_i$ is not diagnosable with respect to the low-level model.

Therefore on any trajectory that starts at $t = t_0$, after some finite time the low-level diagnoser reaches an $F_i$-certain state. This implies that the high-level diagnoser too will reach an $F_i$-certain state. It follows from high-level diagnosability and assumption (B) that this will happen in a bounded time. Hence the low-level diagnoser (which is equivalent to the high-level diagnoser) enters an $F_i$-certain state in a bounded time. This contradicts the assumption that $F_i$ is not diagnosable with respect to the low-level model. □

**Example 4.1 - Water Tank (Cont’d)**

Using Table 4.2, the reader can verify that the partition $\pi = \{B_1, \ldots, B_{16}\}$, with

- $B_1 = \{x \mid q = q_1, h = 0\}$
- $B_{10} = \{x \mid q = q_7, h = 0\}$
- $B_2 = \{x \mid q = q_2, h \in [0, L]\}$
- $B_{11} = \{x \mid q = q_7, h = L\}$
- $B_3 = \{x \mid q = q_3, h \in [L, U]\}$
- $B_{12} = \{x \mid q = q_8, h \in [0, L]\}$
- $B_4 = \{x \mid q = q_3, h = U\}$
- $B_{13} = \{x \mid q = q_9, h \in [L, U]\}$
- $B_5 = \{x \mid q = q_4, h = U\}$
- $B_{14} = \{x \mid q = q_{10}, h = U\}$
- $B_6 = \{x \mid q = q_5, h = U\}$
- $B_{15} = \{x \mid q = q_{11}, h = [U, h_m]\}$
- $B_7 = \{x \mid q = q_6, h \in (L, U]\}$
- $B_{16} = \{x \mid q = q_{12}, h \in [L, U]\}$
- $B_8 = \{x \mid q = q_6, h = L\}$
- $B_9 = \{x \mid q = q_1, h = L\}$
Figure 4.14: Example 4.1. High-level model.

satisfies assumptions (I) and (II), and hence, belongs to $\Pi^*$. The high-level model, reachability transition system and diagnoser are given in Fig. 4.14, Table 4.3 and Fig. 4.15. The low-level and high-level diagnosers produce identical estimates for the system's condition in response to the system's output sequence. Note that the trajectories of the hybrid system satisfy assumptions (A) and (B). According to the high-level model, the valve failure can be diagnosed if it happens while the system is at $B_7$. This is in accordance with the low-level model.

In general, it is very difficult to verify whether a given partition $\pi$ satisfies property (II) (or even whether $\Pi^* \neq \emptyset$). This makes the construction of high-level models difficult.

In the following, we introduce another method for building high-level models which is based on partitions that only have to satisfy property (I) and assumption (B). These DES models are significantly easier to construct. We will show that the
<table>
<thead>
<tr>
<th>Block #</th>
<th>Output-adjacent blocks (output)</th>
<th>Block #</th>
<th>Output-adjacent blocks (output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,12 ( (H_0, C_2) )</td>
<td>10</td>
<td>12 ( (H_0, C_2) )</td>
</tr>
<tr>
<td>2</td>
<td>3,13 ( (H_1, C_2) )</td>
<td>11</td>
<td>12 ( (H_0, C_2) )</td>
</tr>
<tr>
<td>3</td>
<td>5,14 ( (H_2, C_3) )</td>
<td>12</td>
<td>13 ( (H_1, C_2) )</td>
</tr>
<tr>
<td>4</td>
<td>5 ( (H_2, C_3) )</td>
<td>13</td>
<td>14 ( (H_2, C_3) )</td>
</tr>
<tr>
<td>5</td>
<td>6,15 ( (H_2, C_4) )</td>
<td>14</td>
<td>15 ( (H_2, C_4) )</td>
</tr>
<tr>
<td>6</td>
<td>7 ( (H_1, C_4) )</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>9 ( (H_0, C_1) ) / 15 ( (H_2, C_4) )</td>
<td>16</td>
<td>15 ( (H_2, C_4) )</td>
</tr>
<tr>
<td>8</td>
<td>9 ( (H_0, C_1) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2,12 ( (H_0, C_2) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Example 4.1. Reachability transition system for the DES model.

Figure 4.15: Example 4.1. High-level diagnoser.
condition estimates generated by the resulting high-level diagnosers will be more conservative; however, the diagnosers may still be used for fault diagnosis and for verifying diagnosability. Here we are taking an approach similar to that proposed in [16] by Kurshan et al. for developing a semi-algorithmic method for verifying properties of digital circuits.

Usually, it is very difficult to calculate \( X \), the reachable part of the state space of a hybrid system. However, determining a superset of \( X \) might be easier. Let \( X^* \) denote one such superset; hence \( X \subseteq X^* \subseteq Q \times \mathbb{R}^n \). Let \( \hat{\pi} \leq \ker \lambda \land \ker \kappa \) and \( P_\pi : X^* \rightarrow X^*/\hat{\pi} \) be the canonical projection. Based on this partition, let us define the finite-state Moore automaton \( \hat{G} = (\hat{X}, \hat{\Sigma}, \hat{\delta}, \hat{X}_0, \hat{Y}, \hat{\lambda}) \), with \( \hat{X} \), \( \hat{\Sigma} \) and \( \hat{Y} \) as the state, event and output sets. \( \hat{X} := X^*/\hat{\pi} \) and \( \hat{\Sigma} \) is a singleton; i.e., \( \hat{\Sigma} = \{\hat{\sigma}\} \), for some \( \hat{\sigma} \). Let \( \hat{X}_0 := PX_0 \) be the set of initial states, \( \hat{\delta} : \hat{X} \times \hat{\Sigma} \rightarrow 2^{\hat{X}} \) the transition function and \( \hat{\lambda} : \hat{X} \rightarrow \hat{Y} \) the output map. For \( \hat{x}_1, \hat{x}_2 \in \hat{X} \), let \( \hat{x}_2 \in \hat{\delta}(\hat{x}_1, \hat{\sigma}) \) if and only if there exist \( x_1 \in \hat{x}_1 \) and \( x_2 \in \hat{x}_2 \) and a trajectory of finite duration (satisfying the hybrid system’s dynamics) from \( x_1 \) to \( x_2 \) lying in \( \hat{x}_1 \cup \hat{x}_2 \). Note that since \( \hat{\pi} \) does not necessarily satisfy property (II), the block \( \hat{x}_2 \) may not be reachable from every state in \( \hat{x}_1 \). For all \( \hat{x} \in \hat{X} \), \( \hat{\lambda}(\hat{x}) := \lambda(x) \) for any \( x \in \hat{x} \). Similarly define \( \hat{\kappa} : \hat{X} \rightarrow \kappa \) according to \( \hat{\kappa}(\hat{x}) := \kappa(x) \) for any \( x \in \hat{x} \). Since \( \hat{\pi} \leq \ker \lambda \land \ker \kappa \), \( \hat{\lambda} \) and \( \hat{\kappa} \) are well-defined. \( \hat{G} \) is a high-level model for \( H \). In this automaton, a transition from a state \( \hat{x}_1 \) to another state \( \hat{x}_2 \) indicates the existence of trajectory from block \( \hat{x}_1 \) to block \( \hat{x}_2 \) lying entirely in \( \hat{x}_1 \cup \hat{x}_2 \).

**Lemma 4.5** Suppose \( x \Rightarrow x' \) with \( x, x' \in X^* \) (\( x' \) is output-adjacent to \( x \)). Then \( P_\pi x \Rightarrow P_\pi x' \); i.e., in \( \hat{G} \), the block \( P_\pi x' \) is output-adjacent to \( P_\pi x \).

**Proof.** Similar to that of Lemma 4.1, with \( \overline{G} \) and \( \overline{X} \) replaced by \( \hat{G} \) and \( \hat{X} \). \( \square \)

Define the canonical projection of a subset \( z \subseteq X \) to be \( P_\pi z := \cup\{P_\pi x \mid x \in z\} \). Let \( \hat{D} \) be the high-level diagnoser designed based on \( \hat{G} \), with \( \hat{z}_0 := P_\pi z_0 \) as its initial

---

\(^5\lambda \) and \( \kappa \) were earlier defined on \( X \). Here we have assumed that their domain is even larger and includes \( X^* \). This is typically the case. The reachable part of the state set is usually a subset of the domains of the output and condition maps.
state estimate. Let \( \hat{z}_k \) denote the state of \( \hat{D} \) following the generation of the output symbol \( y_k \).

**Theorem 4.6** Let \( z_k \) and \( \hat{z}_k \) denote the states of the low-level diagnoser \( D \) and high-level diagnoser \( \hat{D} \), respectively. Then \( \kappa(z_k) \subseteq \hat{\kappa}(\hat{z}_k) \).

**Proof.** By definition \( P_\pi z_0 = \hat{z}_0 \).

\[
P_\pi z_1 = P_\pi (z_0 \cap \lambda^{-1} \{ y_1 \}) \\
= P_\pi z_0 \cap P_\pi (\lambda^{-1} \{ y_1 \}) \\
= \hat{z}_0 \cap \lambda^{-1} \{ y_1 \} \quad \text{(because } \hat{\pi} \leq \ker \lambda) \\
= \hat{z}_1.
\]

Now we prove that for \( k \geq 1 \) if \( P_\pi z_k \subseteq \hat{z}_k \) then \( P_\pi z_{k+1} \subseteq \hat{z}_{k+1} \).

Let \( x_{k+1} \) be an element of \( z_{k+1} \), i.e., \( x_{k+1} \in z_{k+1} \). Then there exists \( x_k \in z_k \) such that \( x_k \Rightarrow x_{k+1} \). Hence by Lemma 4.5, \( P_\pi x_k \Rightarrow P_\pi x_{k+1} \). From \( P_\pi z_k = \hat{z}_k \) it follows that \( P_\pi x_k \in \hat{z}_k \). Also \( \hat{\lambda}(P_\pi x_{k+1}) = \lambda(x_{k+1}) = y_{k+1} \). As a result, \( P_\pi x_{k+1} \in \hat{z}_{k+1} \). This shows that \( P_\pi z_{k+1} \subseteq \hat{z}_{k+1} \). Finally, \( \kappa(z_k) = \bar{\kappa}(P_\pi z_k) \subseteq \hat{\kappa} (\hat{z}_k) \) since \( \hat{\pi} \leq \ker \lambda \).

The theorem states that \( \hat{D} \) produces more conservative estimates for the system's condition than \( D \). Hence, \( \hat{D} \) is less accurate than \( D \). If \( \hat{D} \) enters an \( F_i \)-certain state, so will \( D \) (but not vice versa).

Let us suppose assumption (B) holds for \( \hat{\pi} \), namely, if the projection of a low-level trajectory in \( X \) on the high-level state space \( X/\hat{\pi} \) is a single state \( \hat{x} \), with \( \hat{\delta}(\hat{x}, \hat{\sigma}) \neq \emptyset \), then the trajectory must be of finite duration. Since calculating \( X \) is difficult, we may verify assumption (B) for the projection of the trajectories in \( X^* \) on \( X^*/\hat{\pi} = \hat{X} \) which is sufficient to guarantee assumption (B) (for the trajectories in \( X \)).

**Theorem 4.7** If a failure mode is diagnosable with respect to the high-level model \( \hat{G} \), then it must be diagnosable with respect to the low-level hybrid model \( H \).

**Proof.** Similar to the (If) part of the proof of Theorem 4.4, except that \( \kappa(z_k) = \bar{\kappa}(\bar{z}_k) \) should be replaced with \( \kappa(z_k) \subseteq \hat{\kappa}(\hat{z}_k) \). The diagnosers are not equivalent; however, if the high-level diagnoser is at an \( F_i \)-certain state, so is the low-level diagnoser. \( \square \)
As a result, if a failure mode is diagnosable with respect to the high-level model, then it is certainly diagnosable with respect to the low-level model and can be diagnosed using \( \hat{D} \), the high-level diagnoser. Therefore the high-level model may replace the low-level one for the purpose of fault diagnosis. But if a failure turns out to be undiagnosable at the high level, it may still be diagnosable at the low level. In this case, the analysis based on the high-level model is inconclusive. At this point, we may replace the existing partition with another (perhaps finer) partition. An analysis based on the resulting diagnoser may show that the failure is diagnosable. If, on the other hand, the failure turns out to be undiagnosable at the high level, then we may try another partition.

In general, the failure may be undiagnosable even at the low level. In these cases, information from the high-level model (e.g., fault-indeterminate cycles) may assist us in proving the undiagnosability of the failure mode at the low level by providing clues about trajectories that may lead the low-level diagnoser into fault-uncertain states.

Based on the above discussion, we introduce the notion of weak consistency.

**Definition 4.8** The low-level and high-level models of the system are said to be **weakly consistent** (for the purpose of fault diagnosis) if and only if

- the estimates for the system's condition generated by the low-level diagnoser are subsets of the estimates given by the high-level diagnoser \( \kappa(z_k) \subseteq \hat{\kappa}(\hat{z}_k) \),
  
  and

- if a failure mode is diagnosable with respect to the high-level model (Def. 2.4), then it must be diagnosable with respect to the low-level model (Def. 4.3). \( \Box \)

**Example 4.2** -
Consider the hybrid system depicted in Fig. 4.16. In this system, \( \Sigma_f = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\} \)
and $\Sigma_e = \{ F \}$. Also
\[
\lambda(q_1, (\xi_1, \xi_2)) = \lambda(q_2, (\xi_1, \xi_2)) = \begin{cases} y_1 & \text{if } \xi_1 < 0, \\ y_2 & \text{if } \xi_1 \geq 0, \end{cases}
\]
\[
\lambda(q_3, (\xi_1, \xi_2)) = y_3,
\]
\[
\lambda(q_4, (\xi_1, \xi_2)) = y_4.
\]

The trajectories of $\xi$ in $\mathbb{R}^2$ in the normal and faulty modes are shown in Fig. 4.17.
If in modes $q_1$ and $q_2$ we partition $\mathbb{R}^2$ according to $\mathbb{R}^2 = B_1 \cup B_2$, with $B_1 = \{(\xi_1, \xi_2) \mid \xi_1 < 0\}$ and $B_2 = \{(\xi_1, \xi_2) \mid \xi_1 \geq 0\}$, then the resulting high-level model $\hat{G}_1$ will be as shown in Fig. 4.18. Here $X^* = (\{q_1\} \times B_1) \cup (\{q_2\} \times B_1) \cup (\{q_2\} \times B_2) \cup (\{q_3\} \times \mathbb{R}^2) \cup (\{q_4\} \times \mathbb{R}^2)$. In this figure, the output at each high-level state is shown. According to $\hat{G}_1$, the failure is not diagnosable since in both normal and faulty modes, the system can generate the output sequence $\cdots y_1 y_2 y_1 y_2 \cdots$.

Now let us use a finer partition for $\mathbb{R}^2$ (in modes $q_1$ and $q_2$): $\mathbb{R}^2 = B'_1 \cup \cdots \cup B'_4$, with $B'_1 = \{(\xi_1, \xi_2) \mid \xi_1 < 0, \xi_2 < 0\}$, $B'_2 = \{(\xi_1, \xi_2) \mid \xi_1 < 0, \xi_2 \geq 0\}$, $B'_3 = \{(\xi_1, \xi_2) \mid \xi_1 \geq 0, \xi_2 < 0\}$ and $B'_4 = \{(\xi_1, \xi_2) \mid \xi_1 \geq 0, \xi_2 \geq 0\}$. The high-level $\hat{G}_2$ is given in Fig. 4.19. In this case, $X^* = (\{q_1\} \times B'_1) \cup \cdots \cup (\{q_1\} \times B'_4) \cup (\{q_2\} \times B'_1) \cup \cdots \cup (\{q_4\} \times B'_4) \cup (\{q_3\} \times \mathbb{R}^2) \cup (\{q_4\} \times \mathbb{R}^2)$. According to $\hat{G}_2$, and in agreement with the hybrid system, in the faulty mode the system only generates one of the two output sequences $\cdots y_1 y_2 y_3 y_1 y_2 y_3 \cdots$ or $\cdots y_1 y_4 y_2 y_1 y_4 y_2 \cdots$. Therefore the failure is high-level, hence, low-level diagnosable. Note that in Fig. 4.19, the event $\alpha$ corresponds to transition from $B'_3$ to $B'_4$ in the mode $q_2$ when $0 \leq \xi_1 < 0.4$ and $\xi_2 = 0$. This transition, however, never happens in the hybrid system because that part of the boundary between $B'_3$ and $B'_4$ is not reachable in the mode $q_2$. \[\square\]
Figure 4.19: Example 4.2. The high-level DES $\hat{G}_2$. 
In the high-level model $\hat{D}$, $\hat{x}_2 \in \delta(\hat{x}_1, \delta)$ for some $\hat{x}_1, \hat{x}_2 \in \hat{X}$ if there exists a trajectory from an $x_1 \in \hat{x}_1$ to some $x_2 \in \hat{x}_2$, lying entirely in $\hat{x}_1 \cup \hat{x}_2$. Verifying this property (i.e., existence of the trajectory) is significantly simpler than property (II) of the partitions in $\Pi^*$. In fact, this property should be investigated only on the boundary between $\hat{x}_1$ and $\hat{x}_2$; specifically, we have to see if a trajectory crosses the boundary.

For example, suppose $\xi = (\xi_1, \xi_2) \in \mathbb{R}^2$ and $Q = \{q\}$. Furthermore, assume that $\mathbb{R}^2$ has been partitioned into rectangular blocks (Fig. 4.20). We would like to know if there exists a trajectory from $B_1$ to $B_2$ in Fig. 4.20. Let $\dot{\xi}_1 = f^1_q(\xi_1, \xi_2)$. If there exists $\xi_2 \in [a, b]$ such that
\[
\dot{\xi}_1 = f^1_q(c, \xi_2) > 0, \tag{4.4}
\]
then certainly a trajectory crosses the boundary from $B_1$ to $B_2$. One way of checking Eq.(4.4) is to obtain the supremum and infimum of $f^1_q$ on the boundary. If $f^1_q$ is monotone in $\xi_2$ (this is true for LTI systems where $f^1_q(\xi_1, \xi_2) = a_1 \xi_1 + a_2 \xi_2$, for $a_1, a_2 \in \mathbb{R}$), then $\min\{f^1_q(c, a), f^1_q(c, b)\} \leq f^1_q(c, \xi_2) \leq \max\{f^1_q(c, a), f^1_q(c, b)\}$. This method for checking boundary crossing can be used in the more general case of $\xi \in \mathbb{R}^n$, especially if the components of $f_q$ are monotone in the components of $\xi = (\xi_1, \ldots, \xi_n)$. It is evident that checking boundary crossing is easier than verifying property (II).

We may also study the sensitivity of the high-level model with respect to changes in the parameters of the functions $f_q$ and external disturbance and as a result, see whether the weak consistency of the high-level model is robust with respect
to parameter changes and disturbance. Let us go back to Fig. 4.20. Assume that 
\( a = (a_1, a_2) \in \mathbb{R}^2 \) and \( w \in \mathbb{R} \) represent system parameters and disturbance input, respectively. Then the dynamics of \( \dot{\xi}_1 \) can be described in more detail with:

\[
\dot{\xi}_1 = f_q^1(\xi_1, \xi_2, w; a_1, a_2).
\]

Suppose \( a_1 \in [a_{1,\text{min}}, a_{1,\text{max}}], a_2 \in [a_{2,\text{min}}, a_{2,\text{max}}], \) and \( |w(t)| \leq w_{\text{max}} \) for all \( t \). In order to check \( \dot{\xi}_1 > 0 \) on the boundary between \( B_1 \) and \( B_2 \), one has to obtain the supremum and infimum of \( f_q^1 \) (as a function of \( \xi_1, \xi_2, w, a_1 \) and \( a_2 \)) on the boundary. If \( f_q^1 \) is monotone in all of its parameters (for example, in LTI systems), then this supremum and infimum can be easily calculated using the extreme values of the parameters.

As an example, consider the DES model shown in Fig. 4.14 which is consistent, hence weakly consistent, with the low-level model. In this automaton, the transition \( B_2 \) to \( B_3 \) indicates that there must be a trajectory from \( B_2 \) to \( B_3 \) (In fact, \( B_3 \) is reachable from all of the states in \( B_2 \) since \( \pi \in \Pi^* \)). We want to know to what extent the crossing of the boundary from \( B_2 \) to \( B_3 \) is robust with respect to the changes in \( q_0 \), the input flow. On the boundary between \( B_2 \) and \( B_3 \):

\[
\dot{h} = \frac{q_0}{A} - \frac{k}{A} \sqrt{L}.
\]

If \( q_0 \) fluctuates, as long as \( q_0 > k\sqrt{L} \), \( \dot{h} \) remains positive and the boundary \( h = L \) can be crossed.

A similar sensitivity analysis for the partitions in \( \Pi^* \) in order to verify the robustness of the consistency of the corresponding high-level models would be extremely difficult (though possible in principle).
Chapter 5

Conclusion

5.1 Summary

In this thesis, we study fault diagnosis using discrete-event models. It is assumed that the plant under supervision is operational and that no test inputs are to be used; diagnosis has to be based on measurements and perhaps, information about the input commands sent to the plant.

First, we propose a state-based framework for fault diagnosis in systems that can be modelled as finite-state automata. It is assumed that the state set of the system can be partitioned according to the system’s condition (failure status). The diagnoser (fault detection system) is a finite-state machine that takes the output sequence of the system as input and generates at its output an estimate of the condition of the system. This approach is relatively simple and straightforward. The system and the diagnoser do not have to be started at the same time. Furthermore, no information about the state or even the condition of the system before the initialization of the diagnoser is required. We also introduce a model reduction scheme with polynomial time complexity for reducing the computational complexity of designing the diagnoser which, in the worst case, has exponential time complexity. We study the issue of diagnosability of failures. In our framework, a permanent failure is diagnosable if it can be detected and isolated after the occurrence of at most a bounded number of events following the occurrence of the failure and initialization of the diagnoser. We
derive necessary and sufficient conditions for failure diagnosability.

We extend our framework to fault diagnosis in timed discrete-event systems by incorporating timing information. This improves the accuracy of diagnosis. In a timed discrete-event system, the sequence of events occurring in the system is described with respect to the ticks of a global clock. We present a new method for using timing information which does not require estimate updates at clock ticks; instead, it relies on the number of clock ticks between consecutive output symbols in the output sequence, and on “predictions” for the system’s condition. This significantly reduces on-line computing requirements, and in many cases, the size of the diagnoser. Time-diagnosability in our framework is also discussed and necessary and sufficient conditions for time-diagnosability are obtained.

Next we study the cases in which the DES model used for fault diagnosis is a simplified model for a more complex hybrid automaton. Here an important issue is whether the simplified high-level model (DES) contains enough information about the plant for the purpose of fault diagnosis. In order to formalize the above question, we first extend our framework for fault diagnosis in DES to hybrid systems. Then we introduce the notion of “consistency” between the high-level (DES) and low-level (hybrid) models. A high-level model is called consistent with the low-level model if (i) the corresponding high-level and low-level diagnosers generate identical estimates for the system’s condition, and (ii) any failure mode is diagnosable with respect to the high-level model if and only if it is diagnosable with respect to the low-level model. A set of sufficient conditions for consistency is presented. Moreover, we introduce a more relaxed notion of consistency, called “weak consistency”, based on which we develop a semi-algorithmic method for obtaining suitable high-level DES models for low-level hybrid systems.

Whether our methodology for fault diagnosis is suitable for application in a system or not, depends on two factors: availability of a model for the system and the complexity of the problem.

Our framework is model-based and therefore, requires a description of the system in its normal and all faulty modes. This makes the technique suitable for cases where
the dynamics of the system are (at least to some extent) understood. Building the model is time consuming; however, the effort can be justified in part by the fact that the model can be used as system requirements specification [21]. Besides, in case of future revision in the system’s design, the diagnostic system can be updated simply by revising the system model.

Note that if the system develops an unforseen failure which was not included in the model, then the diagnoser may still be able to detect the failure based on the discrepancy between the model and observations. In general, in these cases, failure detection can not guaranteed (unless a failure model is available).

A challenge associated with methods based on automata is the reduction of computational complexity. In this thesis, we have addressed this problem in several places. In fact, our work on fault diagnosis primarily deals with two issues: complexity and diagnosability. In the following section, we provide some suggestions for future research on the problem of complexity. Interestingly enough, while appearing as a “demon”, complexity is the prime motive for the study of systematic approaches!

In this thesis, we attempt to provide a new framework for solving a class of fault diagnosis problems. In any real-world system, depending on failure types and the information available about the system, a combination of techniques is typically used to arrive at a solution for the diagnostic problem. Our methodology is meant to be one of the tools in the designer’s toolbox.

5.2 Future Research

Throughout the thesis, we used different examples to illustrate our approach. These examples were simplified, small-scale versions of real-world problems. In order to assess the methodology properly, we need to apply our technique to real systems. As with any technique based on discrete-event models, computational complexity is the most important obstacle that has to be overcome. While the schemes developed in Section 2.4 and Chapter 3 are helpful in reducing complexity, in (larger) real problems, they will not be enough unless they are coupled with techniques that exploit system
architecture for reducing complexity. For example, in a system comprising several subsystems, for the diagnosis of one of the failure modes of a subsystem, a detailed model of the subsystem might be enough, while for another failure, reduced models of a number of the subsystems might be required. A hierarchical decentralized approach would be extremely useful here. In [7] and [8], the idea of decentralized fault diagnosis has been explored.

In this work, we focused on passive diagnosis. Active diagnosis, in which input commands are issued by the diagnoser for testing the system and detecting failures, is another interesting area of research. Here the problem is to find a systematic way of generating input commands suitable for diagnosing failure modes. Active diagnosis of finite-state machines has been studied extensively [18]. In [22], it has been examined in the context of control systems in a state-based framework. Integrated approaches to diagnosis and control have been studied in [5] and [31] in an event-based framework, where the goal is to design a controller that, in addition to satisfying design specifications, renders the system under supervision diagnosable. The resulting diagnoser can only diagnose failures that happen after the diagnoser is initialized. It would be interesting to build on the results of [22], [5] and [31], and develop a practical methodology for active diagnosis of control systems in a state-based framework.

Upon the detection and diagnosis of a failure mode, the control system has to accommodate the failure. Failure accommodation has not been studied in this work but is an important problem for future study. In this problem, fault diagnosis and supervisory control become closely interconnected and will add much more complexity to the analysis.

Our results in Chapter 4 are the first steps towards a practical theory for the verification of consistency in hybrid systems and construction of consistent high-level models. As with other problems in formal verification, obtaining a general systematic solution is extremely difficult (if not impossible). However, it is possible to enlarge the class of systems that can be formally analyzed by developing semi-algorithmic methods, along with useful mathematical and software tools. Application of the techniques developed in Chapter 4 to real-world systems is a suitable starting point.
towards the above goal and should be explored further.

Another issue that we merely touched upon in Chapter 4 is the robustness of (weak) consistency with respect to modelling uncertainty and external disturbance. This, too, is an important matter which is left for future research.
Bibliography


[12] W. Hamscher, L. Console and J. de Kleer, Eds., Readings in Model-Based Diag-

computation of supremal elements in system theory,” SIAM J. Control Optim.,


[15] IEEE Transactions on Automatic Control, Special Issue on Hybrid Control Sys-


[17] M. Larsson, “Diagnosis and analysis of diagnosis properties using discrete event
dynamic systems,” in Proc. 37th IEEE Conf. Decision Contr., Tampa, FL, USA,

[18] D. Lee and M. Yannakakis, “Principles and methods of testing finite state


