A Compositional Learning Diagnoser for Discrete Event Systems

Tarek Noaman Bekir

A Thesis in

The Department of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

March 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Tarek Bekir**

Entitled: **A Compositional Learning Diagnoser for Discrete Event System**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair

Dr. Rastko Selmic

_____ External Examiner

Dr. Youmin Zhang

_____ Examiner

Dr. Rastko Selmic

_____ Supervisor

Dr. S. Hashtrudi Zad

Approved by _____

Dr. Yousef Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2023

_____

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

A Compositional Learning Diagnoser for Discrete Event System

Tarek Bekir

In this thesis, the design of model-based fault diagnosis systems for Discrete Event Systems (DES) is studied. The correct performance of a model-based diagnoser depends on the accuracy of the plant DES model used in its design. The behavior of this nominal model may differ from the actual plant behavior (i.e., the true model) due to various reasons such as modeling errors, modeling simplifications and coding errors. The difference between the nominal model and the true model may result in observations (i.e., sensor readings) that are unexpected by the diagnoser, resulting in "discrepancy."

In the literature, "learning diagnosers" have been proposed that in cases of discrepancy add transitions to the nominal DES model to account for the unexpected behavior. It is assumed that the nominal and true models have the same number of states, and their difference is in the transitions.

In general, every discrepancy can be explained by different sets of additional transitions, each representing a hypothesis. To narrow down the list of hypotheses, the principle of parsimony is used - giving preference to less complex hypotheses. After a set of hypotheses is generated, using future observations, this set is narrowed down. In some cases, the set may have to be expanded.

In this thesis, a new approach to learning diagnoser is introduced that takes advantage of the structure of the plant to generate hypotheses. This is meant to reduce the number of hypotheses and to generate hypotheses that are more likely to correctly explain the discrepancies. Specifically, the proposed method takes advantage of the fact that models are built incrementally from component models and their interactions. Hence, in the learning process, new transitions (to explain discrepancies) are added to the component models and their interactions (rather than to the complete flat model of the plant).

This results in a compositional approach to learning. In this thesis, a framework for the compositional approach is presented and the learning rules and the corresponding algorithms are developed. A case study from process control is used to illustrate the proposed approach.

Acknowledgments

I am highly appreciating and feeling very grateful for my supervisor Dr. Shahin Hashtrudi Zad for his continuous guidance, valuable advice, and kind support throughout my work on this research as well as his effort to overcome any difficulties during the hard times of the pandemic period of COVID-19.

I would also love to thank my family that surrounded me with their support and love, especially my wife Marwa for her patience and encouragement, and my sons Abdulmalek and Abdullah who were very proud of my efforts during my studies.

My deep thanks to my elder brothers Ahmed and Mohamed, who guided me in the very first steps in my life and they were and still are my honest counselors, and my endless support.

Lastly, as I will be forever in dept to my parents for their love, kindness, and dedicated prayers. I wished that they lived to share this moment with me. I dedicate this work with my sincere gratitude to their pure souls.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modelling a Discrete Event Systems (DES) is very important step and a corner stone when dealing with the design of supervisory control and fault diagnosis systems. Our focus in this thesis is directed to automata. Petri nets and automata are two examples for the modelling categories used to study DES [1].

The chosen model category will guide the design process towards the particular design technique. The accuracy of a model is one of the significant aspects that strongly affects the quality of design of systems. A model-based design process of a diagnoser could be prone to incomplete models. This model incompleteness usually appears as a result of a lack of full information about the components' behavior. In addition, model incompleteness could result from the simplification of some component models (e.g., ignoring some transitions or faults that are very unlikely to happen). However, in the long run and in critical applications such as aerospace and industrial processes, the absence of such transitions in the model could severely impact the diagnosis or degrade the quality of the diagnoser.

Due to the importance of model correctness and completeness, vast research work has been carried out for the identification of the missing transitions in incomplete models. In these approaches, observations (e.g., sensor readings) that are not consistent with the plant DES model are used to generate hypotheses about the missing transitions. In this thesis, we aim to develop a learning approach that can generate hypotheses that are more likely to accurately explain the inconsistency between observations and the plant model.

In our approach of this thesis, we aim to identify the missing transitions at the component level. This is based on the fact that true models are constructed from the synchronous product of the constituent automata models of the components. This approach leads to the benefit that once a missing transition is identified at the component level, the correction will subsequently propagate throughout the model and correct many other missing transitions in the composed model.

1.1 Literature Review

This section presents a literature review of research work related to this thesis.

1.1.1 DES Modelling

The nature of a system dictates the technique by which the system can be modelled. In continuous variable systems that are usually represented in terms of differential or difference equations, the state is a continuous vector of values, and the state space is a continuum of these vectors of states. On the other side, in discrete event systems (DES), the change of a variable to another value or from a range of values to another range is associated with an "event", e.g., from low to medium and from medium to high. In continuous systems, state change is driven by time while in discrete event systems, state change is driven by events and therefore, they are called event-driven systems [1].

Modelling discrete event systems typically comes as a result of the abstraction of their real time continuous counterpart systems. This abstraction appears in terms of defining events signifying changes in the levels of the process. For example, in the case of continuous flow rate, defining low flow or high flow events instead of monitoring the full range of the flow could be looked at as an abstraction [1], [3], [51].

A massive body of work has examined modelling techniques of discrete event systems. In general, the two popular and competing techniques are the automata and the Petri nets (PN) [4],[5]. In the literature, various types of automata have been discussed and presented where the modelling methods depend on the characteristics and dynamics of the system. Deterministic automata represent a DES having a single deterministic transition from a state to another by an event [1]. Contrarily, non-deterministic automata represent DES with uncertainty in transitions due to unobservable events, or when a single event leads to multiple different states. Timed automata could be constructed by incorporating timing information along with the events[1],[6] and has some applications in fault diagnosis [7] and supervisory control [8].

Stochastic automata are used to model DES with probabilistic nature where the stares are random and was used to study unpredictable machines failures and sharing resources [1],[9],[10]. Fuzzy logic is incorporated in fuzzy DES to define transition function that results in a membership value to a state based for a given event [11], [12] and is used to study the uncertainty of some applications as problem scheduling, string matching, and database systems [14]. In large DES systems, Hierarchical Finite State Machines (HFSM) divide the DES to small segments called D-hollons each of which represents a stage

of the plant. This approach is used on fault diagnosis where the focus will be on the running D-hollon instead of considering the entire plant model [15][16][17].

Therefore, and in terms of modelling, the chosen approach of the DES is crucially important for the application that uses it. Two major domains that use the DES model are the Fault diagnosis as in [18][16][17][21] and the Supervisory control as in [1] [22].

1.1.2 Observers and Diagnosers

Fault detection and isolation have a crucial importance in practical systems. Extensive research has been carried out in the domain of fault diagnosis. The proposed approaches differ in the model or in the underlying technique for diagnosis. In the literature, there are several approaches for fault diagnosis such as using rough set theory [23], applied on distribution line [24] and the fault diagnosis of incomplete models [25]. Also, the approach of fault tree is used for fault detection and diagnosis of Printed Circuit Board (PCB) [26]. In this thesis we will focus on the fault diagnosis using automata model that is based mainly on DES observers and diagnosers.

In a real system, some events could take place that are not measured and hence are unobservable and drive the system from one state to another. For example, these events are related to some faulty conditions in the plant. An observer [1],[18],[19] is able to give an estimate for the state or the group of states at which the system reaches after an event occurs. This is with taking into consideration the possible unobservable events that might have occurred.

An observer could be the core building block of a diagnoser where after having a state estimate, a mapping function could estimate the normal/faulty condition of the system [20][3]. These conditions could reveal important knowledge about whether the system is behaving normally, or it is in a certain faulty condition [20][3].

In [18] and [19], fault diagnosis using automata model is presented and explored. In this approach, the events set is portioned to normal events and groups of faults. This approach presented an event-based diagnoser that receives the observable events generated as inputs and gives the system condition as output.

While the aforementioned work tackled the systems modeled as standard automata, another version concerned with systems modeled as Moore output automata is discussed in [3],[20] and [27]. In this work, the states are partitioned into groups based on the system condition being in normal or in one of the faulty modes. Consequently, this approach uses a state-based diagnoser that takes as inputs the output labels generated by the system and evaluates the state estimates, then provides the estimated system condition

as output. In [21] an application of such event-based approach has been introduced along with a case study in power grids. A diagnoser that combines both the event-based and the state-based was introduced in [28] as an event-state based diagnoser for timed automata.

In [29], an overview of research work is presented that encompasses a vast amount of literature on the diagnosis methods in DES and groups them from four classification aspects. The first aspect is based on the fault compilation being either off-line or online. The modelling formalism such as petri nets, automata, state charts, and hierarchical state machines is used as the second aspect. The third classification aspect presents the fault representation based on the models that could be event-based, state-based, or fault-tree models. The decision structure is the fourth aspect of this classification that presents three architectures of centralized, decentralized and distributed.

In the literature, some other forms of diagnosis analysis were conducted for specific problems. For example, the robust fault diagnosis and diagnosability are introduced in [30],[31] respectively to deal with stochastic DES where the true model belongs to a set of probabilistic automata. A robust diagnoser is introduced that detects the fault occurrence with no false alarm and with reducing the misdetection rate. Another example is the research work in [32] that introduces the synchronous diagnosis which is based on observing the fault-tree of system components. This is to avoid the large number of states generated by composition of numerous components. The approach is extended to introduce synchronous codiagnosability in a decentralized structure.

In fact, in real-world systems with complexity, not all the faults of the system are diagnosable. A range of undiagnosable faults could be still considered diagnosable and called I-diagnosable if the fault could be detected and isolated after executing the so-called indicator observable event. Otherwise, the fault is considered unobservable [18], and [19].

Partial diagnosable discrete event systems were also studied in [33] to evaluate quantitatively the diagnosability of systems by introducing two indicators. The first is the “diagnosable degree” that briefly gives the weight ratio of a diagnosable fault to all faults. The second indicator is the “diagnosable depth” starting from the root fault to the lowest fault level in the fault model.

1.1.3 Incomplete DES Models

The DES model incompleteness problem has been examined extensively. This problem is concerned with cases where the existing knowledge could be insufficient to model the system completely. This incompleteness in turn affects the accuracy of applications such as fault diagnosis or supervisory control

and therefore needs to be addressed. The incompleteness could result from the abstraction of the real continuous processes that does not sufficiently capture all the dynamics of the system, or it could arise from model simplification [3] [27]. The problem of model incompleteness is addressed in the literature by several approaches that are mainly based on data collection from the system and then using this data to build or complete a model.

Discrete event system identification is an approach that resembles the notion of system identification of the continuous system. It does not require any prior knowledge of the system components [34],[35] and the system could be considered as a black box. In this approach (e.g. [36]), the data collected from a real time system could be used by the identification algorithm to build up an automaton model that could simulate the observed behavior of the system, then it is in fault detection. A recent technique for the identification of DES called Deterministic Automaton with Outputs and Conditional Transitions (DAOCT) that aims to do fault detection is introduced in [37] and [38]. The data is collected to build an automaton representing the fault-free behavior. The model is compared with system observations (i.e., sensor readings) to detect discrepancies and isolate faults.

An extension of the DAOCT approach that incorporates timing information is called the Timed Automaton with Outputs and Conditional Transitions (TAOCT) and is introduced in [39].

Rather than using complete black box identification techniques, a hierarchical approach of two levels was proposed in [40], where the higher level is built by the expert knowledge while the lower is obtained by identification.

1.1.4 Learning in DES

The notion of learning is widely used in the domain of artificial intelligence and machine learning. The basic idea is to start with having a set of data which is used for the learning process using a certain machine learning technique to construct a trained model to be used later for prediction. In the supervised learning, the set of training data includes inputs and outputs so that the trained model can predict the output of any new input. The non-supervised learning has no outputs in the training data and the trained model is used to classify new inputs based on their structure [41], and [42].

The notion of learning has been incorporated in the learning of DES models. The research work in this domain has some general features or aspects. One of the main aspects is the use of online or offline learning algorithms to construct automaton model. Another feature is whether it begins with a completely unknown model, or it begins with a partially known model. In addition, another aspect is whether the

resulting automaton is deterministic or non-deterministic. Offline learning algorithms generally have a set of positive and negative examples respectively for the strings to be accepted or rejected to provide a minimal automaton [43].

An online setup of Angluin's L* algorithm is introduced in [44] to tackle the case of when we start with a completely unknown system. In this method, a set of examples called Teacher and a learning algorithm called Learner are discussed. This approach is employed in the identification of deterministic automata. The L* algorithm of identification is used in a learning-based application of supremal nonblocking supervisor as in [45] and is used also in [46] for a synthesis of permissive supervisor for Markov Decision processes. Also, in [49] deterministic Moore automata are learnt from input and output traces. Furthermore, the NL* algorithm is used for non-deterministic automaton identification in [43] and it is also utilized in the learning of partially known DES in [13].

Since the diagnoser itself is an automaton, the learning of automata was employed in learning the diagnosers. In [47], the proposed learning approach aims to build the diagnoser incrementally from collecting the information of the unknown DES plant in observation tables.

The authors of [3] and [27] consider the problem of fault diagnosis using DES model. There it is assumed that the model used for diagnosis, called the nominal model, may be missing some transitions. The number of missing transitions is not known but an upper bound on the number is assumed available. Furthermore, it is assumed that the nominal model has all the states of the plant's true model. Inconsistency between plant observations and diagnoser is called discrepancy. In case of discrepancy, hypotheses for the missing transitions are proposed to amend the nominal model and resolve the discrepancy. To limit the number of hypotheses, the Parsimonious Covering Theory (PCT) is used.

The PCT that is presented in [48] with the parsimony principle stating that if there are two given explanations where all other factors are equal, it is preferable to use the less complex. Thus, PCT is used in [3], and [27] to construct a learning diagnoser that aims to detect the faults as well as it contributes to completing and repairing the model.

The approach in [3] and [27] uses the flat model of the DES plat which is typically very large. This leads to a large number of hypotheses which may result in the missing transition never been identified. To improve the speed and accuracy of the learning process, it would be useful to take advantage of any knowledge that we have about the structure of the system.

1.2 Objectives and Contributions

1.2.1 Objectives

In this thesis we consider the design of learning diagnoser. Here discrepancies between observations and the nominal model are used to generate hypotheses to amend the nominal model. In order to improve the speed and accuracy of the learning process, we will take advantage of the fact that models for systems are typically built by combining the models of the components and their interactions. Thus, the missing transitions should be added to the model of individual components and interactions (rather than to complete plant model).

1.2.2 Contributions

The contributions of this thesis could be listed in the following items.

- This thesis introduces a new framework to formalize modeling uncertainty that takes advantage of the structure of the system.
- The thesis develops a learning algorithm to use discrepancies between observations and nominal model and generate hypotheses to correct the nominal model.
- The learning model is applied to a case study from process control to identify a missing transition in one of the interaction models on the process.

1.3 Thesis Outline

In chapter 2, we review the required background material for this thesis starting from defining DES with related constituent topics of automata, language, operations, then we introduce observers and diagnosers. Chapter 3 introduces the problem along with its framework by defining the system block diagram components. The components are DES block, nominal model and observer block, Learning algorithm block and the diagnoser block. Then it gives the problem statement followed by two sets of experiments: the first is missing transitions with observable events and the second is missing transitions with unobservable events. In chapter 4, the design of evet-state observer is introduced with its update laws followed by defining the discrepancy symptoms. The mathematical formulation of the learning algorithm is presented, then the computational complexity is discussed. A case study is introduced in chapter 5 to apply the learning algorithm to a process having incomplete model. Chapter 6 lists the accomplishments and contributions. Finally, chapter 7, summarizes the thesis and states the scope of applicability and the limitations and proposes the future work.

Chapter 2

Background Overview

2.1 Discrete Event Systems

The Discrete Event Systems abbreviated as DES are the systems that could be modeled in terms of their states change driven by events. The events could be regarded as an abstraction of the change of the values of the continuous systems where the state variable changes from a discrete value to another or from a range to another, e.g., from close to open or from low to high. The state space becomes a set of states where the transition from a state to another is driven by a discrete event. Such type of systems could be represented as automata or petri nets, where in the study we will focus on automata of Finite State Machines (FSM).

2.2 Languages and Automata

2.2.1 Language

Starting with the basic constructing element of the automata which is the Symbol. While it could be regarded as a letter in the natural language, the symbol represents an event in the model of discrete event system. An alphabet denoted as Σ is a finite set of symbols, and it could be seen as the set of events that can occur in a certain DES. A sequence of symbols (events) formed by the symbols of the language Σ takes the form $\sigma_1 \sigma_2 \sigma_3 \ldots. \sigma_n$ for $n \geq 1$ and $\sigma_i \in \Sigma$, $1 \leq i \leq n$. The sequences are called strings or words. Therefore, for a practical point of view, we can consider the language denoted by L that the DES model can speak means the sequence of events that the system can generate.

For an alphabet Σ, the set of all possible and finite sequences of symbols (events) is denoted as Σ^+.

The word or string that has no symbols or events is called empty string or empty sequence which is denoted by ε.

Σ^* denotes the set of all finite sequences of events in the Σ with including the empty sequence ε.

Thus, $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$.

The length of a string S in the number of symbols in this string and is denoted as |S|, where if S is an empty string i.e., $S = \varepsilon$, then $|S| = 0$. If we consider a string S is comprised of three substrings t,u,v such that $S=tuv$, then t is called a ***prefix*** and v is called a ***suffix***.

2.2.2 Operations on Languages

Consider L and M are two languages defined over the alphabet Σ.

The **intersection** of the two languages is defined as follows:

$L \cap M := \{s \mid s \in L \text{ and } s \in M\}$, is the set of strings that belongs to both L and M.

The **union** of the two languages is defined as follows:

$L \cup M := \{s \mid s \in L \text{ or } s \in M\}$, is the set of strings that belongs to either L or M.

The **concatenation** of two languages L and M is the set of strings that results from concatenating the strings in L and the strings in M. The concatenation of L and M denoted by LM is defined as follows:

$LM := \{tu \in \Sigma^* \mid t \in L \text{ and } u \in M\}$.

The **complement** of a language L is the set of stings in Σ^* that does not belong to L, and is denoted by L^{CO} and is defined as follows:

$L^{CO} := \Sigma^* - L = \{ s \in \Sigma^* \mid s \notin L\}$.

The **prefix-closure** of a language L denote by $\overline{L}$ is defined as follows:

$\overline{L} := \{S \in \Sigma^* \mid \exists\, t \in \Sigma^* [st \in L]\}$, is the set of all prefixes of all the strings in the language L.

The language is said to be prefix closed if $L = \overline{L}$, where in general, $L \subseteq \overline{L}$.

2.2.3 Automata

The automaton or the Finite State Machine is one of the techniques used to model DES [1], [2]. Generally, the automaton could be represented as the following tuple.

$$G=(X,\Sigma,\eta,x_o,X_m),$$

Where X is the set of states, Σ is the finite set of events of the automaton G, η: $X \times \Sigma \rightarrow X$ is the transition function which is a partial function because it might not be guaranteed that all the events are defined at all the states, x_0 is the initial state, X_m: finite set of marked states where $X_m \subseteq X$.

For the empty string denoted as ε, $\eta\ (x,\varepsilon) = x$. If we considered S as a sequence of symbols (events), the transition function η could be extended such that $\eta\ (x,S\ \sigma) = \eta\ (\eta\ (x,S\ ),\sigma)$.

The language that automaton G can generate is called the **closed language** or the **closed behavior** denoted by $L(G)$ and is defined as follows:

$L(G) := \{s \in \Sigma^* \mid \eta(x_0,s) \neq \phi\}$ where $L(G)$ is a closed language, i.e., $L(G) = \overline{L(G)}$.

The **marked language** of the automaton G denoted by $L_m(G)$ is the set of strings that belongs to the closed language of G and ends with marked states, and is defined as follows:

$L_m(G) := \{s \in L(G) \mid \eta(x_0,s) \in X_m\}$. from the definition we can see that $L_m(G) \subseteq L(G)$.

Definition 2.1 Considering a set Q of elements, the ***cardinality*** of this set is the number of the elements in this set denoted as $|Q|$.

□

For example, $Q = \{q_1, q_2, q_3, \ldots, q_{10}\}$, the cardinality of the set Q is $|Q|=10$.

2.2.4 Deterministic and Nondeterministic Automata

The automaton is said to be deterministic if the transition function η is defined such that it evaluates to only one (determined) state. This means that η: $X \times \Sigma \rightarrow X$. An automaton is said to be nondeterministic if the transition function evaluates to more than one state where η: $X \times \Sigma \rightarrow 2^X$ and 2^X represents the power set of X.

Another aspect of nondeterministic automata is the effect of ε transitions where the empty string could cause the system to evolve from a state to other state(s) rather than to keep the system in the same state,

i.e., for a state $x \in X$, then $\eta(x,\varepsilon) \in 2^X$.

2.2.5 Types of Automata Based on Input and Output Mapping

Basically, there are three types of automata used to model the Discrete Event systems based on mapping the events as inputs and the output labels to the states.

2.2.5.1 Standard Automata (FSM)

For its simplicity, this type of automata is widely used in literature and named as the standard automata or Finite State Machine (FSM). It has the same tuple definition as the given formerly and could be represented graphically by the state flow diagram as shown in the following example.

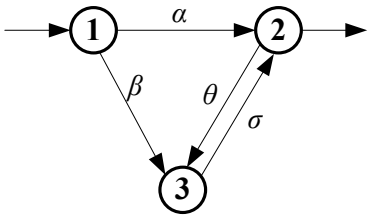


Figure 2.1: Example of Standard FSM

The nodes or circles represent the states, while the symbols over the arcs represent the events. This type of automata can be used to model the systems in which the events can include the enough information to move the system from a state to another. Assume that the system was at the initial state-1, when the event α occurs, the system moves to state-2 and so on. From an operational or practical point of view, the events here include all types of commands (Start/Stop), measurements (High/Low) and any kind of informational or calculated events (Started/Done/Timeout) that cause a transition to another state.

2.2.5.2 Moore Automata

This is the second type of automata in which entering the state is accompanied by emitting an output label. The definition of this type of automata will be extended to include an output map function. Usually in practice, the output labels represent the measurements or readings that indicate entering a certain state. Figure 2.2 shows an example for a Moore automaton with the output labels (y_1,y_2,y_3) indicated over their corresponding states.

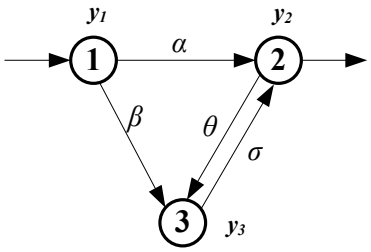


Figure 2.2: Example of Moore Automata with Showing the Events

If the system is initialized at state-1, the output y_1 will be emitted, and if the event α occurs, the system moves to state-2 where the output y_2 is generated and so on. To represent the output mapping of the states to their corresponding output labels, the automaton definition will be extended to include the output map function as given by the following tuple.

$$G=(X,\Sigma,\eta,x_0,Y,\lambda,X_m),$$

Where X is the set of states, Σ: finite set of events, $\eta: X \times \Sigma \rightarrow X$ is the partial transition function, x_0 is the initial state, Y is the finite set of output labels, $\lambda: X \rightarrow Y$ output map function, X_m is the finite set of marked states.

From practical point of view, this type of automata is suitable to model the systems in which it is required to represent the readings of the sensors as output labels while the events that causes the transitions with their causal action on the arcs. For example, the events could be the commands (Start/Stop) while the output labels could be (High/Low).

A simpler version of this automaton is widely used also in the literature where the events are omitted from the arcs since their effect has been already captured by the output labels. For example, as shown in Figure 2.3 with comparison with Figure 2.2, it is known that after the output y_1 is measured at the initial state, the output y_2 will appear after the event α has already occurred. Or let us say for example, the flow became high after the valve has been already opened. Thus, the output labels here sometimes include the enough information of the transition from a state to another.

Note: in the context of this thesis the word "label" will always mean the output label that might be written as just "label" for simplicity.

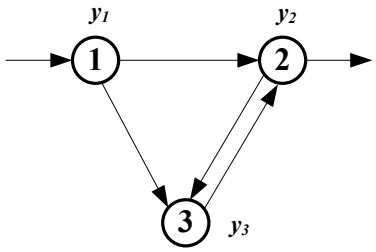


Figure 2.3: Example of Moore Automata with Omitting the Events

2.2.5.3 Mealy Automata

This type of automata with its similarity to Moore automata, the output label is appended to the events on the arcs. Hence, for example in Figure 2.4, it could be interpreted as if the system was at the initial state, then the event α occurred and the output label y_2 is measured, the system is moved to state-2 and so on.

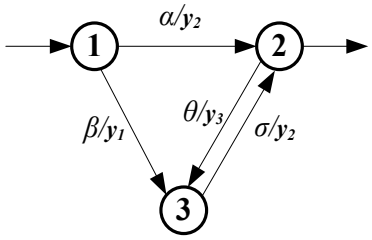


Figure 2.4: Example of Mealy Automata

2.2.6 Partially Observed Systems Modeled as Automata

The notion of having an ε-transition or an empty string could be used to introduce the phenomenon of a DES system that evolves to other states without having an observable event (command or feedback) occurred. In other words, there are some unmeasurable or unobservable events that happened and led to such evolvement and this event could be represented as an ε-transition. Practically, the lack of enough sensor to monitor all the events may lead to such scenarios. But in this case, the evolvement by an ε-transition will add a nondeterminism to the DES model.

Therefore, rather than using the ε-transition with its nondeterminism, another version of the DES systems is introduced to use the unobservable events in lieu of empty string transitions. This version of DES is called the *partially observed DES* and has a deterministic structure that incorporates the unobservable events [1].

2.2.7 Types of Events

While modelling the Discrete Event System as automata, two types of events are to be considered. The first type of events that can occur in the system and could be observed by sensors or by any other monitoring or controlling entity in the system. This type of events is known as the ***observable events***. The alphabet of this type of events is denoted as Σ_O. Examples of these events can include commands to start or stop a pump or the detected measurements of a pressure. On the other side, some events can occur in the system and cannot be observed or measured directly while its effect might appear later on. These events are known as the ***unobservable events***. The alphabet of these events is denoted Σ_{UO}. Examples of these events can include the unobservable failure of a closed valve that becomes stuck in the close position where this could not be measured but its effect on a flow reading can appear later on.

Thus, for a DES modeled as an automaton, the alphabet Σ is the union of these two disjoint sets of events or these two alphabets, i.e., or $\Sigma=\Sigma_O \cup \Sigma_{UO}$.

2.3 Automata Operations

There are so many operations that could be performed on the automata. Here, we will introduce the required operation to be used within the context of this thesis in the following definitions.

2.3.1 Unary Operations

Definition 2.2 for a state and a symbol, the transition function is said to be ***defined*** and denoted as $\eta(x,\sigma)!$ if $\eta(x,\sigma) \neq \phi$

□

This means that at state x, and after the occurrence of the event σ, the automaton can evolve to a set of states evaluated by η that is not an empty set.

Definition 2.3 For an automaton G, a state $x \in X$ is said to be a **reachable state** if there exist a string s

Where, $s \in \Sigma^*$ and $x=\eta(x_0,s)!$

□

Definition 2.4 For an automaton $G=(X,\Sigma,\eta,x_0,X_m)$, and s is a string of symbols, the ***reachable part*** of G written as ***reach*(*G*)** is the sub-automaton G_r with the set of states X_r that is defined as:

$X_r = \{x \in X \mid \exists\, s \in L(G) \text{ and } \eta(x_0, s) = x\}$.

□

The ***reachable part*** of an automaton G denoted as ***reach(G)*** is the sub-automaton G_r of the automaton G where all states are reachable from the initial state x_0, or in other words, it is the sub-automaton after removing the unreachable states.

2.3.2 Binary Operations

Definition 2.5 For the two automata $G_1= (X_1,\Sigma_1,\eta_1,x_{0_1},X_{m1})$ and $G_2= (X_2,\Sigma_2,\eta_2,x_{0_2},X_{m2})$, the product of G1 and G2 written as ***Product*(*G*$_1$,*G*$_2$)** could be defined as follows:

Product*(*G*$_1$,*G*$_2$) = *reach $(X_1 \times X_2,\ \Sigma_1 \cap \Sigma_2, \eta,\ (x_{0_1},x_{0_2}),\ X_{m_1} \times X_{m_2})$.

Where the transition function η is defined as follows:

$$\eta((x_1,x_2),\sigma) := \begin{cases} (\eta_1(x_1,\sigma),\eta_2(x_2,\sigma)) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ and } \eta_1(x_1,\sigma)! \text{ and } \eta_2(x_2,\sigma)! \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Where $x_1 \in X_1$, $x_2 \in X_2$, and $\sigma \in \Sigma_1 \cap \Sigma_2$.

□

For two automata, a symbol or an event is said to be ***common*** if this event or symbol belongs to the languages of both of the two automata while the event is said to be ***private*** to an automaton if it belongs ONLY to the alphabet that specific automaton.

Definition 2.6 For the two automata $G_1= (X_1,\Sigma_1,\eta_1,x_{0_1},X_{m1})$ and $G_2= (X_2,\Sigma_2,\eta_2,x_{0_2},X_{m2})$ we write the ***synchronous product*** of G1 and G2 as ***sync* (*G*$_1$,*G*$_2$)** and could be defined as follows:

sync* (*G*$_1$,*G*$_2$)** = ***reach $(X_1 \times X_2,\ \Sigma_1 \cup \Sigma_2, \eta,\ (x_{0_1},x_{0_2}),\ X_{m_1} \times X_{m_2})$,

where for $x_1 \in X_1$, $x_2 \in X_2$, and $\sigma \in \Sigma_1 \cup \Sigma_2$

$$\eta((x_1,x_2),\sigma) := \begin{cases} (\eta_1(x_1,\sigma),\eta_2(x_2,\sigma)) & \text{If } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ and } \eta_1(x_1,\sigma)! \text{ and } \eta_2(x_2,\sigma)! \\ (\eta_1(x_1,\sigma), x_2) & \text{If } \sigma \in \Sigma_1 - \Sigma_2 \text{ and } \eta_1(x_1,\sigma)! \\ (x_1, \eta_2(x_2,\sigma)) & \text{If } \sigma \in \Sigma_2 - \Sigma_1 \text{ and } \eta_2(x_2,\sigma)! \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

□

This ***synchronous product*** operation is also called the ***parallel composition***.

Definition 2.7 Consider a model that is built by synchronous product of some components, at a certain state x of the entire system's model, the ***state value*** of a component is the specific state of the component at which this component reached when the system reached x .

□

For example, for a system built by the synchronous product from the components V1, V2, F respectively, and reached the state x =(C,O,H), the state value of F is H.

Definition 2.8 The ***ε-Reach*** operation of a state $x \in X$ denoted by ***εR(x)*** will return all the states that could be reached by the empty string ε starting from the state x. Normally, x belongs to $\varepsilon R(x)$.

The ε-Reach could be extended for a set of states $Z \in X$, where the ε-Reach is the union of all the ε-Reach of the member states and could be written as $\varepsilon R(Z) = \bigcup_{x \in Z} \varepsilon R(x)$

□

Definition 2.9 for an automaton G, consider the set of events Σ that could be divided into two disjoint subsets Σ_O, Σ_{UO} such that $\Sigma = \Sigma_O \cup \Sigma_{UO}$, where Σ_O is the set of observable events and Σ_{UO} is the set of unobservable events where, also $\varepsilon \in \Sigma_{UO}$, the ***unobservable reach*** of the state $x \in X$ denoted as $UR(x)$ is given by:

$UR(x) = \{y \in X : (\exists\ s \in \Sigma_{UO}^*)$ and $\eta(x,s) = y\}$.

□

By considering the following example shown in Figure 2.5, the events α_1 and α_2 are unobservable events.

$UR(x_3) = \{x_3\}$

$UR(x_4) = \{x_4 , x_6 , x_7\}$

If $Z=\{x_4 , x_5\}$ then $UR(Z) = UR(\{x_4 , x_5\}) = \{x_4 , x_5 , x_6 , x_7\}$

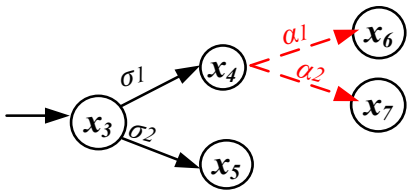


Figure 2.5: Unobservable Reach Example

2.4 Observers and Diagnosers

Since the partially observed DES models contain observable and unobservable events, it will be required to have an observing entity to monitor the progression of the DES automaton while there are unobservable events.

The following example explains the need to employ an observing block in the system to monitor the progression by estimating the states that the system might reach after an observable event occurs.

The observable events (σ_1,σ_2) are represented by arcs with solid lines, meanwhile the unobservable events are represented by dotted lines.

As shown in Figure 2.6, the system is at state x_3 and an observable event σ_1 occurred, so the model says that the system shall progress to state x_4. But maybe there are some transitions in the model after x_4 having unobservable events α_1 and α_2 such that they might happen while they are unmeasurable. Thus, the observer shall estimate that the system is in any of these states $\{x_4, x_{11}, x_{12}\}$.

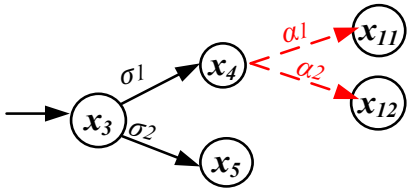


Figure 2.6: Example on the Need for Using an Observer.

2.4.1 Observer

As mentioned formerly, the main purpose of the observer is to estimate the states to which the system might reached after an observable event has occurred considering that unobservable events might occurred.

The observer of a DES is an automaton which is built using the automaton model of the system and evaluates the *unobservable reach* of each observable event in the DES. Then observer automaton will be the version of the DES model that has all the transitions defined by observable events [1].

For a partially observed automaton $G=(X,\Sigma,\eta,x_0,X_m)$ where $\Sigma=\Sigma_O \cup \Sigma_{UO}$, the observer is the automaton $G_{obs}=(X_{obs},\Sigma_o,\eta_{obs},\ x_{0_{obs}},\ X_{m_{obs}})$ that could be computed by the following formulas [1]:

$x_{0_{obs}} = UR\ (x_0)$.

$\eta_{obs}\ (Z,\sigma) =\ UR\ (\{x \in X : (\exists x_z \in Z)\ and\ (x=\eta(x_z,\sigma))\}$.

$X_m = \{Z \in X_{obs} : Z \cap X_m \neq \phi\}$.

The following example shown in Figure 2.7, demonstrates the construction of the observer.

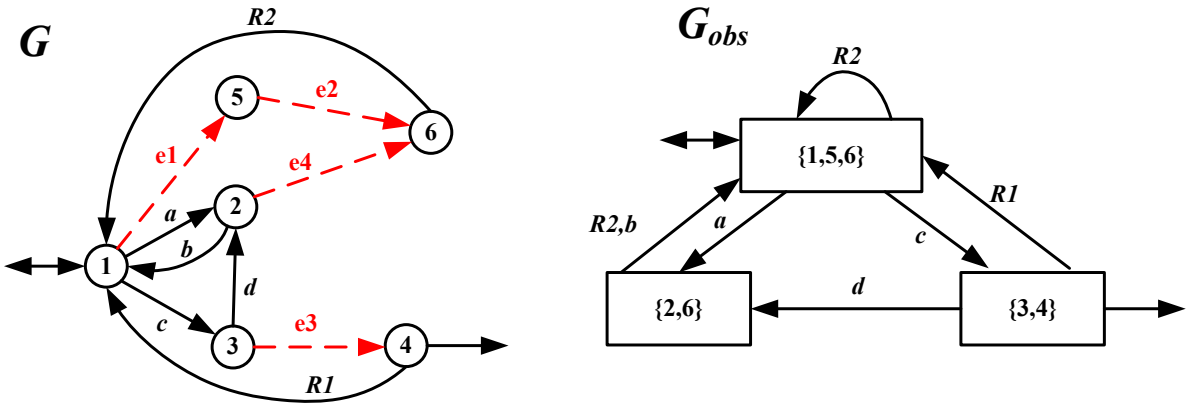


Figure 2.7: Example of Constructing an Observer

$x_{0_{obs}} = UR\ (x_0) = UR\ (1) = \{1,5,6\}$.

At state-1 if the event "a" occurs, the estimated destination will be $UR(\eta(1,a)) = UR(2) = \{2,6\}$

At state-1 if the event "c" occurs, the estimated destination will be $UR(\eta(1,c)) = UR(3) = \{3,4\}$ and so on. The same operation conitues with all the states with their corresponding observable events to construct the remaining transitions in G_{obs}. For $X_m=\{1,4\}$ in the automaton G, thus $X_{m_{obs}}$ is the set of states in G_{obs} that has a nonempty intersection with X_m which are {1,5,6} and {3,4}.

2.4.2 Diagnoser

Faults are considered as a type of unobservable events where it could not be measured directly by sensors, but their occurrence could be diagnosed based on their effect that appears after their occurrence or after some transitions in the system.

The diagnoser is an automaton that is designed based on the current model of the system. It could be seen as an advanced version of the observer where it can do the job of the observer to estimate the current state of the partially observed system. In addition, it can diagnose which unobservable fault(s) event occurred and map it to a system condition.

Two approaches were basically tackled in the literature. The first is the event-based approach introduced in [18] where the system is modeled as a standard FSM. The second is the state-based approach introduced in [20] where the system is modeled as Moore automaton. In this thesis, we will discuss the background review of the state-based diagnoser approach [20].

For a plant modeled as a nondeterministic Moore automaton $G = (X,\Sigma,\eta,x_0,Y,\lambda)$, it is assumed that there are p failure modes denoted as $F_1, \ldots F_P$ and the normal mode denoted as N. The ***condition set*** denoted by $\mathcal{K} = \{N, F_1, \ldots F_p\}$. The set of states X is segmented into subsets of states according to the condition such that $X = X_N \uplus X_{F1} \uplus \ldots \uplus X_{FP}$ where $\uplus$ means the disjoint union. The condition map is denoted as $\kappa(x)$ where $\kappa: X \rightarrow \mathcal{K}$ is a function that maps each state to its corresponding condition based on its membership in which of the segmented subsets of X. The condition map is extended to act on subsets where $\kappa(z) = \{\kappa(x) \mid x \in z\}$.

The diagnoser monitors the plant modeled as a Moore automaton while generating the output labels reflecting the measurements ($y_1, y_2, \ldots$). Based on the generated output label, the diagnoser will estimate the current state. Due to the unobservable events, the diagnoser estimation evaluates to a set of states denoted as z. Then the diagnoser uses the condition map $\kappa(z)$ to decide the condition. Therefore, the diagnoser could be regarded as a unit that receives the output label sequence ($y_1, y_2, \ldots$) to estimate the

system condition in one of the modes in the condition set $\mathcal{K} = \{N, F_1, \ldots F_p\}$.

In this approach, the diagnoser D is defined as a Moore automaton as follows:

$D = (\mathcal{Z} \cup \{\underline{z}_o\}, Y, \xi, \underline{z}_0, \mathcal{K}^\wedge, \hat{k})$ where $\mathcal{Z}$ is the set of this diagnoser states $z_1, z_2, ..$ (Each single state z is a set of states estimated for the diagnosed plant automaton G. The initial state $\underline{z}_0 = (z_0, 0)$ is denoted as a pair because it has a different update law where $z_0 \in 2^X - \{\phi\}$ and usually $z_0 = X$. The set of output labels of the diagnosed automaton G denoted by Y is the event set of the diagnoser D. $\mathcal{K}^\wedge \subseteq 2^{\mathcal{K}} - \{\phi\}$ is the output set of the diagnoser while $\hat{k}: \mathcal{Z} \cup \{\underline{z}_0\} \rightarrow \mathcal{K}$ is the output map function.

The partial transition function is defined as $\xi: \mathcal{Z} \cup \{\underline{z}_0\} \times Y \rightarrow Z$ where the state transition $z_{K+1} = \xi(z_K, y_{K+1})$ is given by:

$$z_1 = z_0 \cap \lambda^{-1}(\{y_1\}).$$

$$z_{K+1} = \{x \mid \lambda(x) = y_{k+1} \text{ and } (\exists\ x' \in z_K : x' \Rightarrow x)\}.$$

Where $x' \Rightarrow x$ denotes that x' is output adjacent to x which means that x is reachable from x' through a path of states in which all the outputs = $\lambda(x)$ but the output at $\lambda(x') \neq \lambda(x)$ and this output $\lambda(x')$ is considered the first output change after the constant output path.

In terms of computational complexity, it is possible to economize the computation by doing reachability analysis. In this analysis, breadth-first search is used to compute the output-adjacent states for each state in the model. The result of this analysis will be stored in a table called Reachability Transition System (RTS) [20]. The following example explains the idea.

Example 2.1: Consider the following automaton as the model of the system.

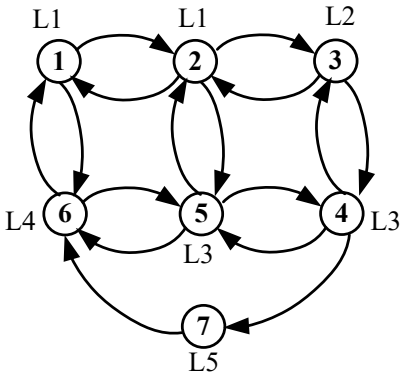


Figure 2.8: Example Automaton for RTS

Table 2.1: Reachability Transition System

State	Output-adjacent states
1	3,5,6
2	3,5,6
3	2,4
4	3,6,7
5	3,6,7
6	1,5
7	6

To explain how the output-adjacent states are computed in the reachability transition system, we consider the state 1, it has two outing transitions. The first from 1 to 2 where the label is still L1 the same as state 1, if we continue to state 3, the label changes to L2, thus this is the first output change on a path ($1 \rightarrow 2 \rightarrow 3$) with the same label reachable from 1. From 2 if moved to 5, this will be first label change of the label to be L3 on the path ($1 \rightarrow 2 \rightarrow 5$), thus 5 is output adjacent. Finally, the second outing transition from 1 to 6, will find a new label at 6 of L4 on the path ($1 \rightarrow 6$) thus, 6 is output adjacent. Therefore, the state 1 has {3,5,6} output-adjacent states that will be substituted in the RTS.

Chapter 3

Problem Formulation

This chapter will discuss the research problem of this thesis to find an algorithm that corrects the incomplete DES automata models. The following sections of this chapter introduce Fault diagnosis framework and the experiments operated to find what is the useful information that could be used to conclude some rules that lead to construct the learning algorithm. The experiments will tackle two types of missing transitions, the missing transitions with observable events and the missing transitions with unobservable events.

3.1 The Fault Diagnosis Framework

Before stating the research problem, it is required to present the framework within which the problem originates, and the proposed learning algorithm will perform. The framework here means the entire block diagram that includes all the entities that comprise the fault diagnosis application of this thesis. This includes the true plant, the model along with the observer that monitors the plant and finally the learning algorithm and its position among all these components.

The importance of introducing the framework before the algorithm details and the formulation is to introduce the functional operation of the algorithm. This functional operation will explain how and when the algorithm is triggered and what is expected to be received as input data and what should be produced as output. Figure 3.1 represents the operation framework in which the proposed learning algorithm should function.

3.1.1 DES Block

The true plant along with the controller is modeled as a discrete event system (DES). In this block, the plant reactions are observed typically by controller commands and measurements in terms of observable events and the generated output labels. These observations (observable events and output labels) will be applied as inputs to the observer block.

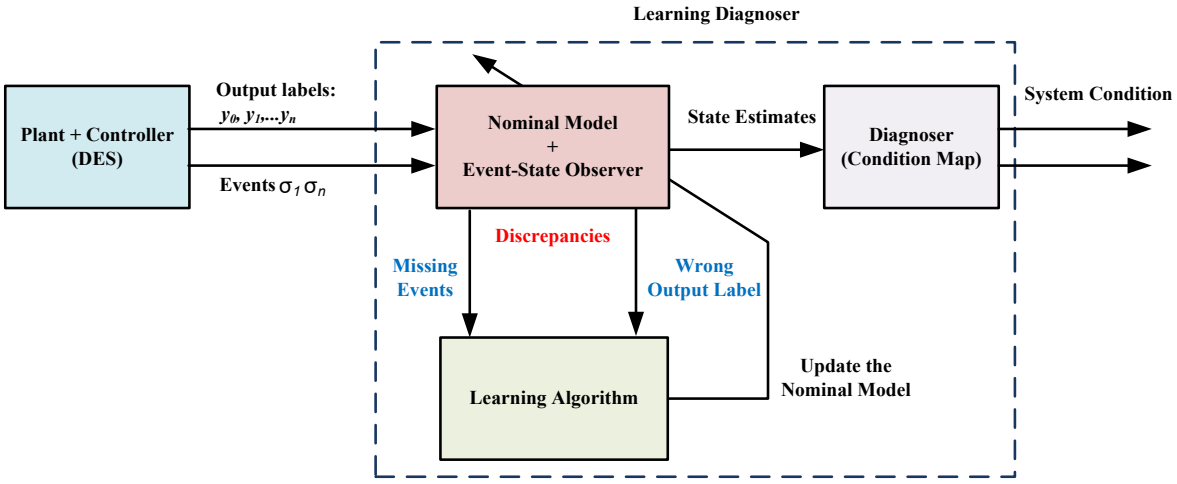


Figure 3.1: Entire Framework Block Diagram

3.1.2 Nominal Model and Observer

Since the systems modeled as DES contain unobservable events, it is required to incorporate an observer to estimate the current state of the system. The observer is designed using the current model of the system which is composed by the synchronous product of the automata of the components and is referred to as the ***nominal model***. The components automata are modeled by the best of knowledge, thus, if there are any missing transitions, in some component will lead to an incomplete model that might behave differently from the plant.

The observer that is designed based on the nominal model estimates the system progression from one state to another taking into consideration the unobservable events. While the observer is monitoring the progression of the system, if there is any new observation (output label or observable event) that could not be explained by the observer and it has an empty set of estimates, this condition is flagged as discrepancy. Then, the discrepancy is forwarded to the learning algorithm for correction. In this thesis, the observer is an event-state observer that uses both events and output labels to estimate the next state and will be discussed in detail in the following chapters.

3.1.3 Diagnoser Block

The diagnoser block receives the state estimates from the observer and it contains a condition map function that maps the estimated states to their corresponding condition either normal or fault. It should be noted that, in the literature as in [20], the typical diagnoser carries out both tasks of being an observer to estimate the states and then maps these estimates to the corresponding condition. In this thesis, the two

tasks are separated, where the estimates are handled by the observer while the condition mapping is handled by the diagnoser.

3.1.4 Learning Algorithm

Recalling that the observer is monitoring the plant by receiving the events and output labels and it aims to estimate the progression from one state to another. If the nominal model was incomplete in terms of missing some transitions, so its behavior does not match the behavior of the real plant, (either it cannot explain the events or the output labels) then, a discrepancy will be flagged. This discrepancy is signaled to the learning algorithm that will collect and analyze all the current information from the true plant and the nominal model. Then it works on correcting the nominal incomplete model by hypothesizing the missing transitions to match the real plant. Such a correction will update the nominal model that in consequence will update the observer. This is indicated by the updating arrow in the block diagram.

3.1.5 Learning Diagnoser

As shown in the block diagram, the three blocks of, the nominal model and observer, the learning algorithm and the diagnoser will comprise the function of the bigger block of the learning diagnoser. This is in fact to introduce the idea of designing a diagnoser that monitors the plant and learns to get updated by detecting any discrepancy in the current nominal model that will be corrected by the learning algorithm. Consequently, the learning diagnoser is updated based on the new learnt nominal model.

3.2 Problem Statement

The DES models represented by automata are constructed by the synchronous product of the components' automata and the interactions automata. In case of lack of knowledge or modeling mistakes, or if the automaton of a certain component is missing some transitions, this in turn will affect the entire synchronous product and the resulting model becomes incomplete. Such model incompleteness would impact the correctness of the application that could be fault diagnosis or supervisory control as examples.

Since the focus of this thesis is on the domain of fault diagnosis, a diagnoser will be used to monitor the plant and to evaluate the states estimation of the model. This estimation is done based on the events and the output labels occurring in the plant. If the model is incomplete, the plant will generate some events or output labels that leads the observer to estimate an empty set of states ϕ. Such an empty estimation of states blocks the diagnoser from evolving to estimate the next states to which the model has moved to and

generates a discrepancy. The discrepancy describes the case where the diagnoser is unable to explain or to estimate which state the system is progressed to because the events and output labels do not match with the current available incomplete model. This thesis tackles the problem of designing a learning diagnoser by finding an algorithm that exploits the current available information in the plant and the nominal model to generate hypotheses for the missing transitions to correct the incomplete models. The correction will be carried out on the level of components to correct the wrongly modeled components that are suspected to be the cause of the discrepancy.

The general assumptions concerned with the problem are as follows:

- The automata of the components are modeled as Moore Deterministic automata and the resulting model is a deterministic Moor automaton.
- The components are modeled with the correct number of states, and they do not miss any state.
- The alphabet of event symbols is well defined, and we are not adding more events.
- The correct number of components are modeled, and we are not missing any component.
- The output map of the model is known.
- The unobservable events are known, and we are not adding new unobservable events.

In the coming sections or chapters, the specific assumptions will be listed individually to cope with the corresponding concerned topic.

3.3 Steps Towards the Solution

In this research, a set of tests and experiments are operated to conclude some rules by which it is possible to formulate a heuristic learning algorithm that can be used to correct the incomplete models.

We start the experiment with an exactly well-defined and known DES model represented by an automaton of the entire model. This model will be referred to in the context of this research as the ***true model***. The second step of the experiment is to deliberately delete some transitions in certain components and redo the synchronous product. Thus, another incomplete version of the model is generated by synchronous product. This resulting automaton will be referred to as the ***nominal model***. The goal is to emulate the situation as if according to our best of knowledge we modeled a system and resulted in the nominal model. But since this model is still incomplete and missing some transitions, then it is required to find a method to detect the missing transitions and hypothesis them to achieve the complete true model.

The two models, the true model automaton and the nominal model automaton will be evolved by some sequence of events such that the true model will generate output labels and progress normally. On the other side, the nominal model will evolve with the same sequence of events as the true model. The response of the nominal model towards the applied sequence of events will be monitored in terms of complying with the sequence of events and generating the same output labels as the true system model or not. In case of any mismatch of the generated output labels or if the nominal model is unable to proceed with the sequence of event, a discrepancy flag shall be raised signifying that a missing transition is detected.

A group of experiments are operated by the formerly explained scenario and some rules were concluded to detect the suspected components and to hypothesize the missing transitions. These rules will be used to formulate a ***learning algorithm*** that is proposed to solve the problem of correcting the incomplete DES models represented by automata.

Owing to the fact that, the DES systems automata might have unobservable events that are not measurable. Therefore, monitoring the evolvement of the DES automata is done by observers or diagnosers to estimate the current state which usually be in form of an estimated set of states.

It should be noted that, during the experiments operated to conclude a learning algorithm, the true and nominal models and the sequence of events are exactly known. Therefore, the observer or diagnoser is not incorporated in the experiments. The evolvement will be done step by step or state by state by comparing the two models instantly against each other. Later on, after concluding the learning algorithm rules, the application of the algorithm on some examples and a case study will incorporate using an observer to estimate the current set of states that the nominal automaton might evolved to. Briefly, the work will be started assuming the model automaton evolves from a single state to exactly a single state. Then the work will be extended where the model evolves from a single to a set of state estimates (by incorporating the observer).

3.4 Introduction to Experiments

3.4.1 Types of Missing Transitions in Automata

This section discusses the possible types of missing transition in the models. As shown in Figure 3.2, the two general types of missing transitions are related to the nature of events being observable or unobservable. These two general types will pursue two different branches in the learning algorithm. In addition, the observable events could be categorized in terms of alphabet as private to a single component

or it could be common between two or more components in the system.

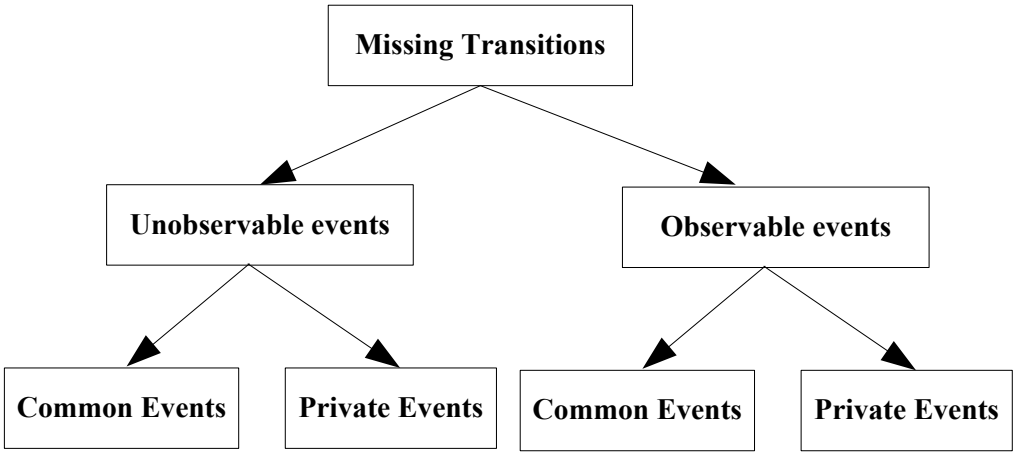


Figure 3.2: Type of Missing Transitions

3.4.2 Steps of Experiments

These are the general steps of operating the experiments:

1- Start with a system having a group of components e.g., valves, pump, motor, etc.

2- Model the components as Moore automata.

3- Perform the synchronous product to get the complete automaton of the true model.

4- Selecting the automaton of one or more components and delete some transitions deliberately.

5- With the modified components, perform the synchronous product to get the automaton of the nominal model that is missing transitions.

6- Apply a known sequence of events on both true model and nominal model and observe the progression.

7- If the nominal model has a different response or fails to proceed with the same sequence of events, a discrepancy is detected.

8- Collect the current information observed at the moment of discrepancy.

9- Analyze the observations to determine the suspected components, sources, and destinations of the missing transitions.

10- Generate the hypotheses in the suspected components (Correct the missing transitions in the components).

11- Perform synchronous product to update the nominal model.

12- Continue the progression of the models with applying events and reject the incompliant hypotheses.

3.4.3 Experiments Preparation and Organization

As mentioned in the steps of the experiments above, some transition will be deleted deliberately, so we organized the experiments based on the following criteria.

- Missing transitions with observable events are divided into two types tested separately. The first is for private events and the second is for common events.
- Missing transitions with unobservable events are tested if it occurs with its effect appears after an observable event and while its effect appears without observable events.
- We start with testing the case of deleting a single transition, then multiple transitions to see the effect of the model structure on the ability to detect and retrieve the missing transitions.
- We test changing the sequence of events from the true system to see its effect changing the observation while the model structure is fixed.
- Throughout all the experiment, the main goal is to make sure that the algorithm is able to correct the missing transitions or to learn about the limitations if there are any.

3.4.4 Definitions

Definition 3.1: The ***Output map*** it is a table that is built by listing all the possible states of the entire model that are comprised by the cartesian product of the states of the individual components along with the corresponding output label as shown in the following example.

Assume that we have a system of cascaded valve and a pump V and P, followed by a flow meter as shown in Figure 3.3. The automaton of V has the states of X_v={C,O} standing for closed and open. The pump automaton has states of X_p={Off,On}. The flowmeter reading will be taken as the output label where it is either High denoted as H or Low denoted as L.

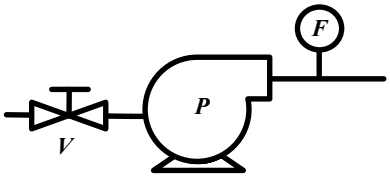


Figure 3.3: Example Automaton for Demonstration

X_v = {C,O}, X_p = {Off,On}

The cartesian product of X_v and X_p is given by $X_v \times X_p$ = {(C,Off),(C,On),(O,Off),(O,On)}. Thus, the output map table is constructed as follows:

Table 3.1: Output Map for the Demonstration Example

V	***P***	***Label***
C	*Off*	*L*
C	*On*	*L*
O	*Off*	*L*
O	*On*	*H*

The first row corresponds to the first state in the cartesian product (C,Off) which will produce a measurement of the Low flow dented by the output label L. similarly, the last row corresponds to the last state (O,On) with a flow reading of High denoted by H.

The output map will be used in the experiments, since building the model from the components is done by the synchronous product which starts with calculating the cartesian product of the states of the components. Then each of the states of the resulting automaton should be assigned an output label according to the definition of the Moore automata. That's why the output map should be prepared for the experiments, and it is assumed that the outputs are known to the designer or the system engineer.

Definition 3.2: The ***Discrepancy State (DS)*** is state of the nominal model at which the discrepancy in event or output label occurs or is detected as mismatch between the behavior of the nominal model and the true mode.

Definition 3.3: The ***Suspected Components (SC)*** are the components suspected of having the missing transition.

Definition 3.4: The ***Last Conforming State (LCS)*** is the last state in the nominal model that was conforming with true model by the applied sequence of events without discrepancy neither in event nor the output label.

Definition 3.5: The ***Last Conforming Label (LCL)*** is the last output label in the nominal model that conforms with the actual received output label from the true model.

Definition 3.6: The ***Actual Label of Discrepancy (ALD)*** is the actual output label when a discrepancy is detected.

Definition 3.7: The ***Expected label*** is the output label generated by the nominal model or component at a specified state.

Definition 3.8: The ***Event of Discrepancy*** is the event causing the nominal model to generate a discrepancy because it is missing at the discrepancy state.

Definition 3.9: The ***Hypothesis Source*** is the state in the component from which the missing transition starts.

Definition 3.10: The ***Hypothesis Destination*** is the state in the component at which the missing transition ends.

3.5 Missing Transitions with Observable Events

This section will present a set of experiments to conclude the building rules of the heuristic algorithm that will be used to recover the missing transitions having observable events.

3.5.1 Assumptions of the Experiments

1- The components e.g., valve or pump are modeled with the correct number of states and are not missing any states.

2- Only some transitions with observable events are missing in the automata of the components.

3- The missing transitions have observable events that are known in the alphabet, and they are not new events to be added to the alphabet.

4- The output map is known, and it is the same for the true and nominal model.

5- The events of faults are unobservable and for simplicity, it is assumed that the unobservable events are considered to be observable, and they will not occur while studying the missing transitions with observable events.

Under the above assumptions, a discrepancy between the nominal model and the actual plant observation is in the form of an observable event that is generated in the plant but is undefined in the nominal model.

3.5.2 Private Events

In this section, the cases of missing private observable events in the nominal model will be discussed. Starting by the system shown in Figure 3.4, the system is comprised of two cascaded valves V1 and V2 followed by a flowmeter F.

The first valve automaton **V1** is a two-state valve of open and closed positions modeled by the automaton V1. The states are:

C: Close. **O**: Open. **SC**: Stuck Close. **SO**: Stuck Open.

The events symbols are defined as follows:

V1C: Close command. **V1O**: Open Command.

V1SC: Stuck Close fault. **V1SO**: Stuck Open fault.

The second valve automaton **V2** is a gradually controlled valve abstracted as three positions of Close, Partially Open and Open modeled by the automaton V2 where the states are:

C: Close. **Po**: Partially Open. **O**: Open. **SC**: Stuck Close. **SO**: Stuck Open.

The events symbols are defined as follows:

V2CT: Close one Turn command. **V2OT**: Open one Turn command.

V1SC: Stuck Close fault. **V1SO**: Stuck Open fault.

The controller automaton **Cont.** is the controller sequence for this system modeled by the automaton **Cont.** The control sequence will start by opening V1 then opening one turn of V2 then open the second turn of V2. For the closing sequence, the controller will close V2 one turn then close V2 the second turn and finally closes V1.

The output labels given by the flowmeter F are:

fL: Low flow **fM**: Medium Flow **fH**: High flow

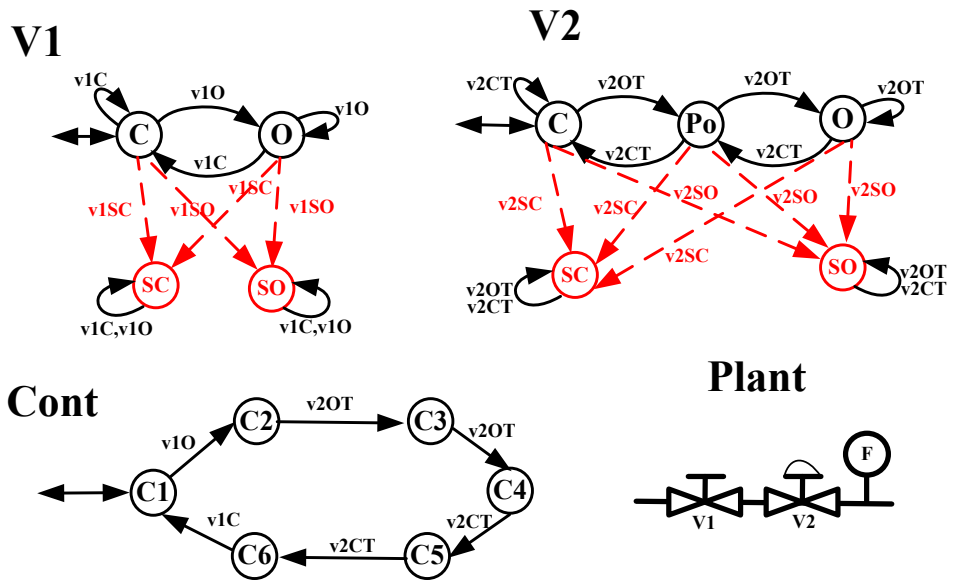


Figure 3.4: Private Events True Model

The output map is given in Table 3.2 below.

Table 3.2: Output Map of Private Events Experimental System

V1	V2	F
C	C	fL
C	Po	fL
C	O	fL
C	SC	fL
C	SO	fL
O	C	fL
O	Po	**fM**
O	O	**fH**
O	SC	fL
O	SO	**fH**
SC	C	fL
SC	Po	fL
SC	O	fL
SC	SC	fL
SC	SO	fL
SO	C	fL
SO	Po	**fM**
SO	O	**fH**
SO	SC	fL
SO	SO	**fH**

The system model will be computed by performing the synchronous product of V1,V2 and Cont

$$G_t = \text{Sync } (V1,V2,\text{Cont})$$

will produce the automaton of the true model or the real plant. Now we will delete the transition with V2CT between O and Po and the transition with V2CT between Po and C as shown by the crossed arcs in Figure 3.5.

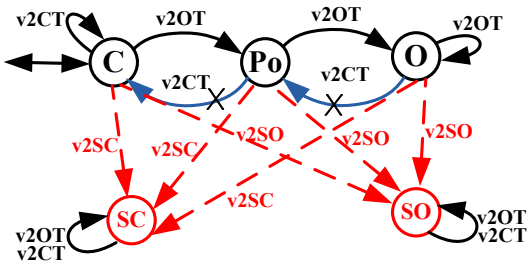


Figure 3.5: Nominal Model of V2

This deletion of transition is to deliberately simulate the problem in which the modeling is done while lacking or missing some transitions (as if this this a modeling or coding error or it is the best of available knowledge currently), so the component V2 is incomplete and will be denoted as V2'.

By reperforming the synchronous product of V1,V2',Cont we will have another incomplete version of the model that is referred to as the ***nominal model.*** This nominal model is regarded practically as the model resulting after a modeling process while missing some information due to any reason and it required to find a method to correct the missing transitions.

The nominal system is

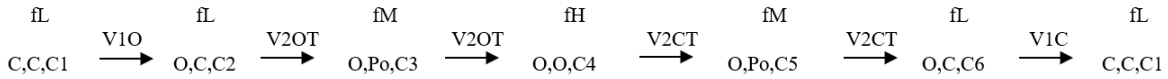
$$G_N = \text{Sync}(V1, V2',\text{Cont})$$

Note: The experiments will be numbered as Experiment-Pn, where P stands for the "private events" while n is the experiment number.

3.5.2.1 Experiment-P1

A sequence of events will be applied to both the complete and the nominal automata. In this example, this sequence of events is generated by the controller "Cont". The following state transition chart depicts the progression of each automaton. As shown below in Figure 3.6, the events' sequence of V1O,V2OT,V2OT,V2CT,V2CT,V1C is applied to both automata. The associated output labels are indicated at the top of each state.

True model:



Nominal Model: Missing V2CT between O and Po in V2

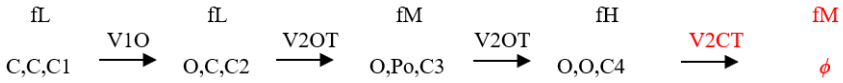


Figure 3.6: P1 True and Nominal Models

The first complete automaton is compatible with the sequence of events as all these events are defined at the states. Meanwhile, the second nominal model reached the state (O,O,C4) where both valves are fully opened, and the output is high flow then after applying the event V2CT is could not progress to another state. This is because this automaton is built by *sync* operation while deleting V2CT in V2 at state O. Hence, when the model reaches the state O,O,C4, the event V2CT is not defined at this state and cannot proceed to another state. This situation is referred to as a **discrepancy** where the nominal model is unable to progress while an event is generated by the real system. Thus, up to here, we simulated a situation in which the system is modeled incompletely and while its progression it has received an event generated by the real system where this event is not defined at the current state of the model. It is required to create an algorithm that can recover the missing transitions. The following procedure will explain how to exploit the current information to hypothesize the missing transitions.

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? Missing Event V2CT

   c. V2CT belongs to the alphabet of V2 and Cont, thus the suspected components ={Cont., V2}

   d. **Last Conforming State**(**LCS**) in the Nominal model:(O,O,C4)

   e. **Actual Label of Discrepancy** (**ALD**) = fM.

2- **Step-2 Analysis:**

   a. Check the first suspected components **Cont**:

   - The state **C4** is the state in **Cont** when the nominal model reached the last conforming state O,O,**C4**.
   - At C4, the event V2CT is found to be defined.

- Then, remove Cont from the suspected components.

b. Check the second suspected components **V2**:

- The state O is the state in V2 when the model reached the last conforming state O,O,C4.
- At O, the event V2CT is found undefined or missing.
- Then, V2 is left in the suspected components.

c. From the Last Conforming State (O,O,C4), **V2 which is the suspected component** was at State **O**.

- Thus, the **source** of missing transition is state O of V2.

Note: This means here, it is required to have a transition that starts from O in V2 with the event V2CT and result in a label of fM.

d. From the output map, list all rows that can have the Actual Label of Discrepancy (ALD)=fM

So, now filtered the output map to limited options in Table 3.3 below.

Table 3.3: Experiment P1: Filtered Output Map

V1	V2	Output
O	Po	fM
So	Po	fM

e. The discrepancy happened due to missing a transition with an observable event.

- When the model reached O,O,C4, the value of the irrelevant component V1 was at O
- Thus, select from Table 3.3, the state(s) that matches this value of V1

Table 3.4: Experiment P1 Filtered Candidates

V1	V2	Label
O	Po	fM

- The remaining row in the output map is the ***candidate expected states*** = (O,Po) where the value of the suspected component **V2** should equal **Po** to have a label = fM.
- Thus, the candidate destination for the missing transition in V2 = {Po}

3- Step-3: Hypothesize:

a. Create a direct transition from the source to the candidate destination on the form:

The suspected component: (Source → Discrepancy event → Destination Candidates).

$V2: O \xrightarrow{V2CT} Po$

4- **Step-4: Update the model**

   a. Re-combine all the components based on the new hypotheses in the suspected components.

Proceed with the system and reject the non-conforming hypotheses if there are any.

Now, the nominal model can progress with the event V2CT at state O,O,C4 to generate the label fM and to proceed to the state O,Po,C5.

3.5.2.2 Experiment-P2

Using the same system for in Experiment-P1 and continuing with the same applied sequence of events, another V2CT command will be applied at the current state. But again, the updated nominal model is unable to progress on since the event V2CT is not defined at the state O,Po,C5.

Nominal model: Missing V2CT between Po and C in V2

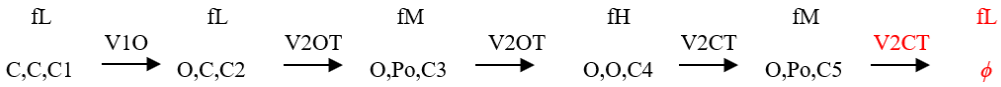


Figure 3.7: P2 Nominal Model

The same procedure given above will be followed to hypothesize the missing transition.

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? Missing Event V2CT

   c. V2CT belongs to the alphabet of V2 and Cont, thus the suspected components ={Cont., V2}

   d. Last Conforming State(LCS) in the nominal model:(O,Po,C5)

   e. Actual Label of Discrepancy (ALD) = fL.

2- **Step-2 Analysis:**

   a. Check the first suspected components **Cont**:

   - The state C5 is the state in Cont when the nominal model reached the last conforming state O,Po,C5.
   - At C5, the event V2CT is found to be defined.
   - Then, remove Cont from the suspected components.

   b. Check the second suspected components **V2**:

   - The state Po is the state in V2 when the model reached the last conforming state O,Po,C5.
   - At Po, the event V2CT is found undefined or missing.
   - Then, V2 is left in the suspected components.

c. From the Last Conforming State (O,Po,C5), **V2 which is the suspected component** was at State Po

- Thus, The source of missing transition in V2 = Po

Note: This means here, it is required to have a transition that starts from Po in V2 with the event V2CT and results in a label of fL.

d. From the output map, list all rows that can have the Actual Label of Discrepancy (ALD)=fL

Table 3.5: Expeiement P2 Output Map

V1	V2	F
C	C	fL
C	Po	fL
C	O	fL
C	SC	fL
C	SO	fL
O	C	fL
O	SC	fL
SC	C	fL
SC	Po	fL
SC	O	fL
SC	SC	fL
SC	SO	fL
SO	C	fL
SO	SC	fL

- When the model reached O,Po,C5, the value of the irrelevant component V1 was at O
- Thus, select the state(s) that matches this value of V1

Table 3.6: Experiment P2 - Filtered Output Map

V1	V2	Label
O	C	fL
O	SC	fL

- Since the discrepancy was due to missing a transition with an observable event, it will be expected that V2 will reach a normal state not a faulty state. This means that the second row in Table 3.6 is not applicable because V2 will be at SC which is a faulty state that is reachable by the unobservable fault events. The remaining row in the table becomes:

Table 3.7: Experiment P2 Filtered Candidates

V1	V2	Label
O	C	fL

- The remaining row in the output map is the ***candidate expected state*** = (O,C) where the value of the suspected component V2 should equal **C** to have a label = **fL**.
- Thus, the candidate destination for the missing transition in V2 = {C}

3- **Step-3: Hypothesize:**

a. Create a direct transition from the source to the candidate destination on the form:

The suspected component: (Source → Discrepancy event → Destination Candidates).

$V2: Po \xrightarrow{V2CT} C$

4- **Step-4: Update the model**

a. Re-combine all the components based on the new hypotheses in the suspected components.

After the above correction, the nominal model becomes the same as the true model and by continuing to apply the sequence of event, the nominal model now can progress without discrepancies.

3.5.2.3 Discussion

The outcome of this experiment is the ability to show that it was possible to recover the missing transitions in the nominal model by using the available information collected from the plant and the nominal model at the occurrence of the discrepancy.

The true automaton model could be regarded as a real plant generating events and outputs. For the sake of performing the tests, the progression of its current states is shown. This was only required in this phase of research for the rules of the learning algorithm. In the practical application of the algorithm, the real system will be a black box or unknown entity which is modeled with the best of available knowledge as the nominal model. After creating the algorithm, the plant works and generates events and outputs. These observations are used to track the state of the plant if a discrepancy occurs, the algorithm will be used to correct the nominal model.

This is in fact analogous to the concepts of learning techniques in machine learning and artificial intelligence. The neural network as an example starts its learning by using a known data set and its corresponding outputs. Once the learning is finished, the learned neural network as an algorithm can deal with the unseen sets of data to predict their outputs.

Therefore, briefly, what is done here in the experiment is to use the known true model along with the

nominal model to drive the learning algorithm, to be used in the correction of the unseen discrepancy cases. In the following, we consider other types of transitions to drive other rules for the learning algorithm.

3.5.3 Common Events

This section will discuss the case of the missing transition pertaining to a common observable event between the components and how to detect the missing event and how to generate hypothesis for model correction.

The following system in Figure 3.8 will be used as an example for the test cases.

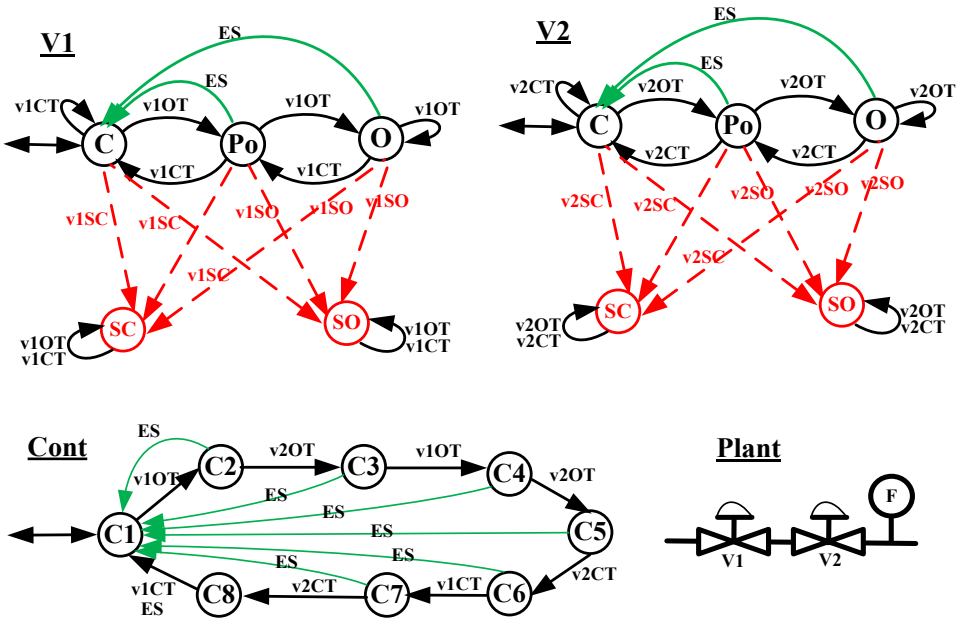


Figure 3.8: Common Events True Model.

The plant is comprised of two valves V1 and V2 each of three states of **closed**, **partially open**, and **open.** They have a **common event** of **ES** that stands for **Emergency Shutoff** that will command both valves to close from any state and is common to the alphabets of V1 and V2. Both valves have the same description as given in Experiment-P1. The controller automaton is shown in Figure 3.8, where the normal operation sequence is V1OT,V2OT,V1OT,V2OT,V2CT,V1CT,V2CT,V1CT.

Note: The experiments will be numbered as Experiment-Cn, where **C** stands for **common** events while n is the experiment number.

3.5.3.1 Experiment-C1

In this experiment, the automaton of the true model is computed by the synchronous product of V1,V2 and Cont.

$Plant_r$ = Synch (V1,V2, Cont)

To have a nominal model emulating a modeling mistake, the event ES will be deleted in V2 between the states Po and C as indicated by the crossed dotted line in Figure 3.9Figure 3.8. Suppose that the system follows its normal sequence, except after the sixth step, instead of V2CT, ES occurs.

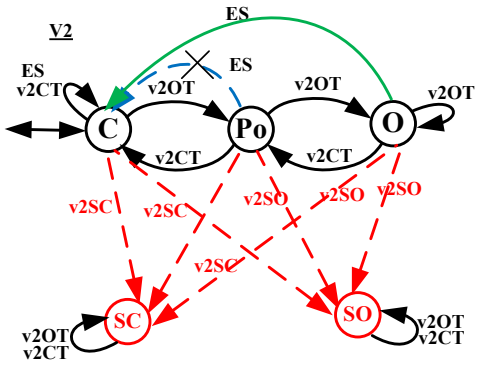
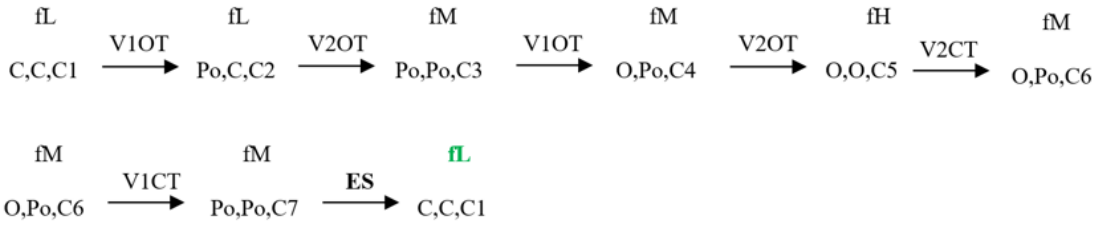


Figure 3.9: V2 Nominal Model

True model:



Nominal Model: Missing ES between Po and C in V2

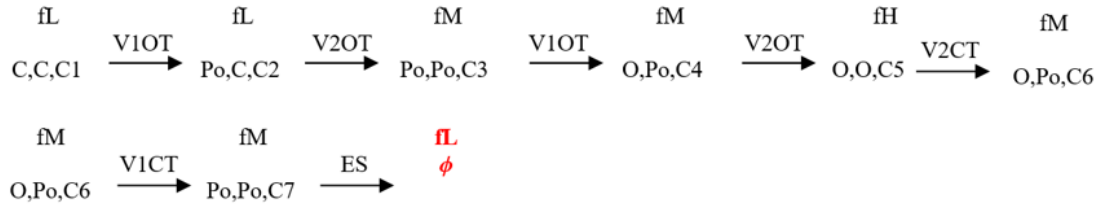


Figure 3.10: Experiment C-1 True and Nominal Models

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? Missing Event ES

   c. ES belongs to the alphabet of V1,V2 and Cont, thus the suspected components ={Cont,V1,V2}

   d. Last Conforming State(LCS) in the Nominal model:(Po,Po,C7)

   e. Actual Label of Discrepancy (ALD) = fL.

2- **Step-2 Analysis:**

   a. Check the first suspected components **Cont**:

   - The state **C7** is the state in **Cont** when the nominal model reached the last conforming state Po,Po,**C7**.
   - At C7, the event ES is found to be defined.
   - Then, remove Cont from the suspected components.

   b. Check the next suspected component **V1**:

   - The state Po is the state in V1 when the nominal model reached the last conforming state **Po**,Po,C7.
   - At Po, the event ES is found to be defined.
   - Then, remove V1 from the suspected components.

   c. Check the next suspected component **V2**:

   - The state **Po** is the state in **V2** when the nominal model reached the last conforming state Po,**Po**,C7.
   - At **Po**, the event **ES** is found **undefined**.
   - Then, **V2** is left in the suspected components.

   d. From the Last Conforming State (Po,Po,C7), **V2 which is the suspected component** was at State Po.

   - Thus, the **source** of missing transition in V2 is the state Po.

   Note: This means that here, it is required to have a transition that starts from **Po** in **V2** with the event **ES** and result in a label of **fL** (as the Actual label of discrepancy).

   e. From the output map, list all rows that can have the Actual Label of Discrepancy (ALD)=fL

Table 3.8: Output Map filtered to the Label fL

V1	V2	Output
C	C	fL
C	Po	fL
C	O	fL
C	SC	fL
C	SO	fL
SC	C	fL
SC	Po	fL
SC	O	fL
SC	SC	fL
SC	SO	fL
Po	C	fL

V1	V2	Output
O	C	fL
SO	C	fL
Po	SC	fL
O	SC	fL
SO	SC	fL

- When the model reached **LCS=(Po,Po,C7)**, and after the event **ES** happened the value of the irrelevant component **V1** should be **C**.
- Thus, select the state(s) that matches this value of **C** in **V1**.

Table 3.9: Filtered Output Map

V1	V2	Output
C	C	fL
C	Po	fL
C	O	fL
C	SC	fL
C	SO	fL

- Since the discrepancy was due to missing a transition with an observable event, the transition should **not** end with a faulty state that is reachable by unobservable event. So, reject the row (C,SC) and (C,SO), i.e., the event ES should take V2 from Po to a normal state which is not one of SC or SO. Table 3.9 is filtered to be:

Table 3.10: Candidate States

V1	V2	Output
C	C	fL
C	Po	fL
C	O	fL

- The remaining row in the output map are the ***candidate expected states*** = {(C,C),(C,Po),(C,O)} and they have the value of the suspected component **V2** could be any state of **{O,Po,C}** to have output = **fL** (as the Actual label of discrepancy) along with **V1=C**.

- Thus, the destination candidates for the missing transition in V2 = **{O,Po,C}**.

3- **Step-3: Hypothesize:**

a. Create a direct transition from the source to the candidate destination on the form:

Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

$H1 = V2{:}Po \xrightarrow{ES} O$

$H2 = V2{:}Po \xrightarrow{ES} Po$

$H3 = V2{:}Po \xrightarrow{ES} C$

4- **Step-4: Update the model**

a. Re-combine all the components based on the new hypotheses in the suspected components. **Three** hypothetical models are generated based on the three hypotheses **H1**,**H2** and **H3**.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the hypotheses incompliant with observations.

After ES, the future progression for the true model and the hypothetical models generated by hypotheses are as follows:

Complete system:

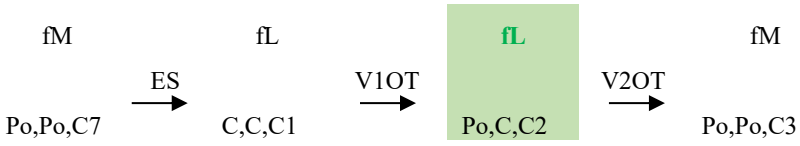


Figure 3.11: Complete System Progression

H1= V2:Po $\xrightarrow{ES}$ O

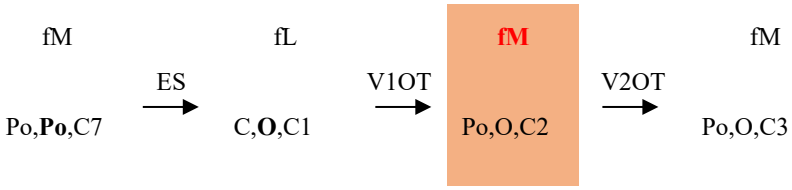


Figure 3.12: H1 Progression

H1 is incompliant with the true model future progression, as it mistakenly predicts fM after **V1OT**. Thus, **H1** is **rejected**.

H2= V2:Po $\xrightarrow{ES}$ Po

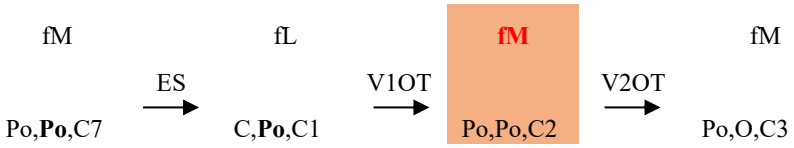


Figure 3.13: H2 Progression

H2 is incompliant with the true model future progression, as it mistakenly predicts fM after **V1OT**. Thus, **H2** is **rejected**.

H3= V2:Po $\xrightarrow{ES}$ C

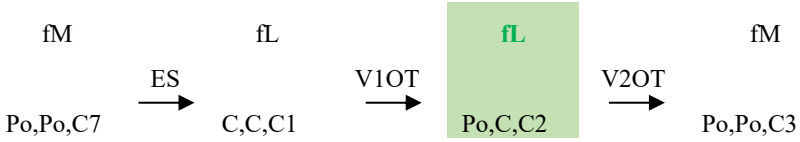


Figure 3.14: H3 Progression

H1 is found compliant with the true model future progression, thus **H1** is **accepted**.

3.5.3.2 Experiment C-2

In this test, still the effect of missing a transition with a common observable event in a single component will be studied but with changing the component position in the control sequence. Here, the same plant as Experiment **C-1** is used, but **V1** will have the common event **ES** deleted between the states **Po** and **C** as shown in Figure 3.15.

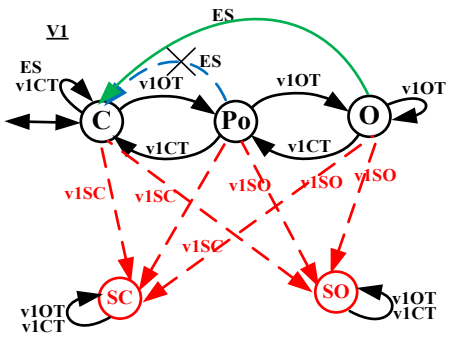
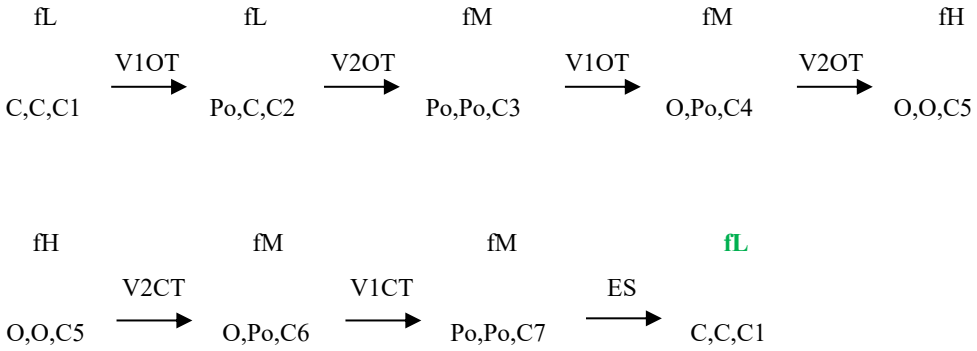


Figure 3.15: V1 After Deleting a Transition

True model:



Nominal Model: Missing ES between Po and C in V1

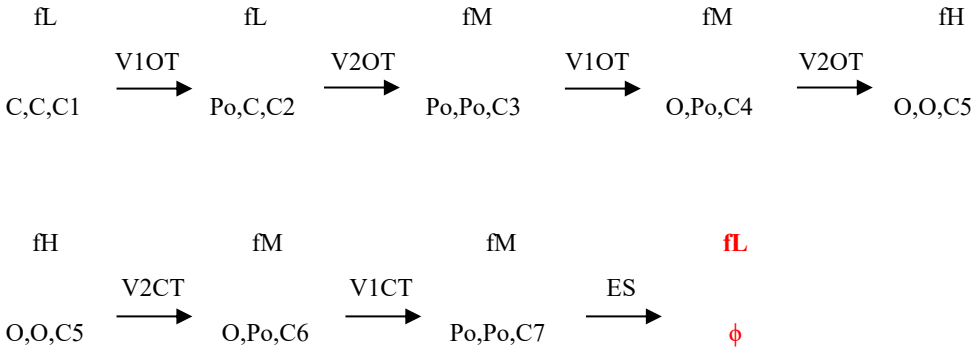


Figure 3.16: Experiment C2 True and Nominal Models

1- **Step-1 Information Collection**
   a. Is there a discrepancy detected? Yes
   b. Type of Discrepancy? Missing Event **ES**
   c. ES belongs to the alphabet of V1,V2 and Cont, thus the suspected components ={Cont,V1,V2}
   d. **Last Conforming State(LCS)** in the Nominal model:(Po,Po,C7)
   e. **Actual Label of Discrepancy (ALD)** = fL.
2- **Step-2 Analysis:**
   a. Check the first suspected components **Cont**:
      - The state **C7** is the state in **Cont** when the nominal model reached the last conforming state Po,Po,**C7**.
      - At **C7**, the event **ES** is found defined.
      - Then, remove **Cont** from the suspected components.

b. Check the next suspected component **V1**:

- The state **Po** is the state in **V1** when the nominal model reached the last conforming state **Po**,Po,C7.
- At **Po**, the event **ES** is found **undefined**.
- Then, is left in the suspected components.

c. Check the next suspected component **V2**:

- The state **Po** is the state in **V2** when the nominal model reached the last conforming state Po,**Po**,C7.
- At **Po**, the event **ES** is found defined.
- Then, remove **V2** from the suspected components.
- From the Last Conforming State (Po,Po,C7), **V1 which is the suspected component** was at State Po.
- Thus, the **source** of missing transition in V1 is from state Po.

Note: This means here, it is required to have a transition that starts from **Po** in **V1** with the event **ES** and result in a label of **fL** (as the Actual label of discrepancy).

d. From the output map, list all rows that can have the Actual Label of Discrepancy (ALD)=fL

Table 3.11: Output Map filtered to the Label fL

V1	V2	Output
C	C	fL
C	Po	fL
C	O	fL
C	SC	fL
C	SO	fL
SC	C	fL
SC	Po	fL
SC	O	fL
SC	SC	fL
SC	SO	fL
Po	C	fL
O	C	fL
SO	C	fL
Po	SC	fL
O	SC	fL
SO	SC	fL

- When the model reached **LCS=(Po,Po,C7)**, and after the event **ES** happened the value of the irrelevant component **V2** should be **C**.

- Thus, select the state(s) that matches this value of **C** in **V2**.

Table 3.12: Filtered Output Map

V1	V2	Output
C	C	fL
Po	C	fL
O	C	fL
SC	C	fL
SO	C	fL

- Since the discrepancy was due to missing a transition with an observable event, the transition should not end with a faulty state that is reachable by unobservable event since the unobservable events were assumed that they did not occur. So, reject the row (SC,C) and (SO,C), i.e., the event ES should take V1 from Po to a normal state which is **NOT** one of SC or SO. The table becomes:

Table 3.13: Candidate States

V1	V2	Output
C	C	fL
Po	C	fL
O	C	fL

- The remaining rows in the output map are the ***candidate expected states*** ={(C,C),(Po,C),(O,C)}. They have the value of the suspected component **V1** could be any state of **{O,Po,C}** to have output = **fL** (as the Actual label of discrepancy) along with **V2=C**.

  Thus, the destination candidates for the missing transition in V1 are {O,Po,C}.

3- <u>Step-3: Hypothesize:</u>

a. Create a direct transition from the source to the candidate destination on the form: Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

$H1 = V1{:}Po \xrightarrow{ES} O$

$H2 = V1{:}Po \xrightarrow{ES} Po$

$H3 = V1{:}Po \xrightarrow{ES} C$

4- **Step-4: Update the model**

a. Re-combine all the components based on the new hypotheses in the suspected component. **Three** hypothetical models are generated based on the three hypotheses **H1**,**H2** and **H3**.

b. Check the future progression of all hypotheses according to the actual labels generated by the real system and reject the incompliant hypotheses.

After ES, the future progression for the true model and the hypothetical models generated by hypotheses as follows:

True Model:

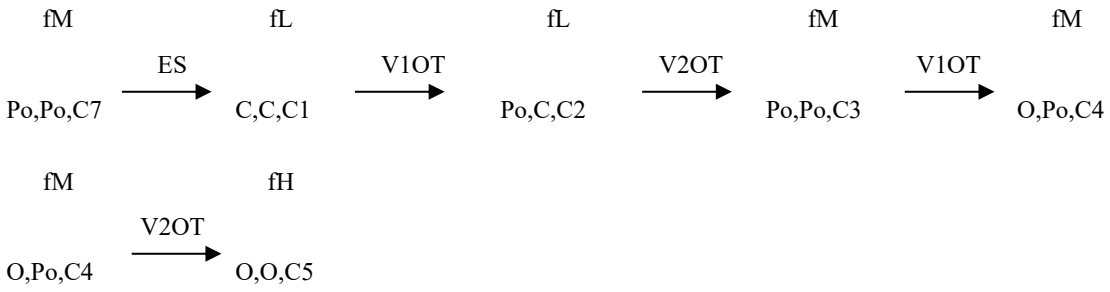


Figure 3.17: True Model System Progression

H1= V1:Po $\xrightarrow{ES}$ O

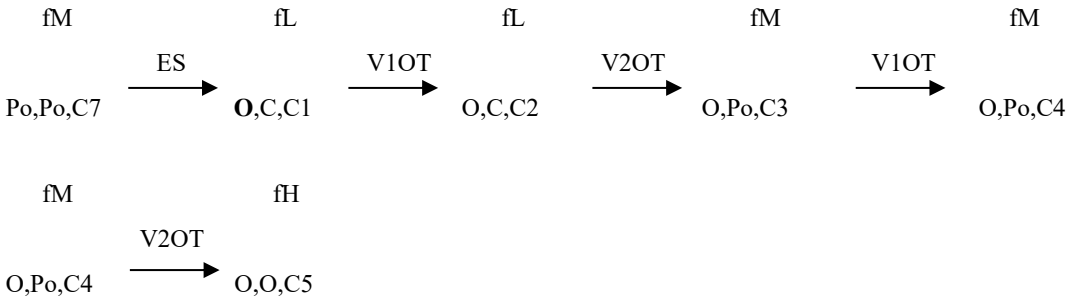


Figure 3.18: H1 System Progression

H1 is compliant with the true model future progression, so it is **kept**.

H2= V2:Po $\xrightarrow{ES}$ Po

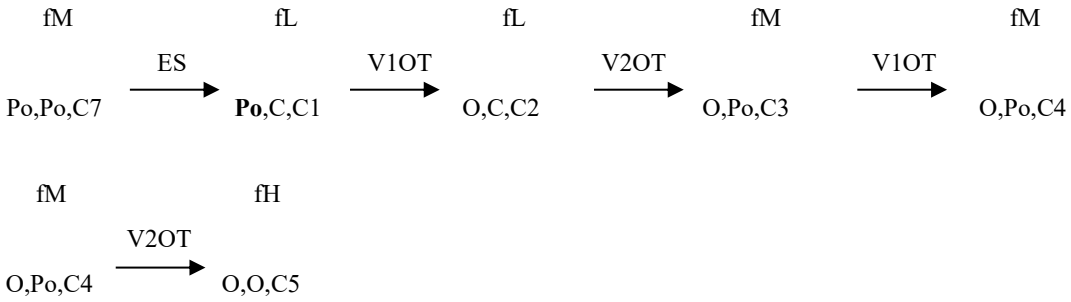


Figure 3.19: H2 System Progression

H2 is compliant with the true model future progression, so it is **kept**.

H3= V2:Po $\xrightarrow{ES}$ C

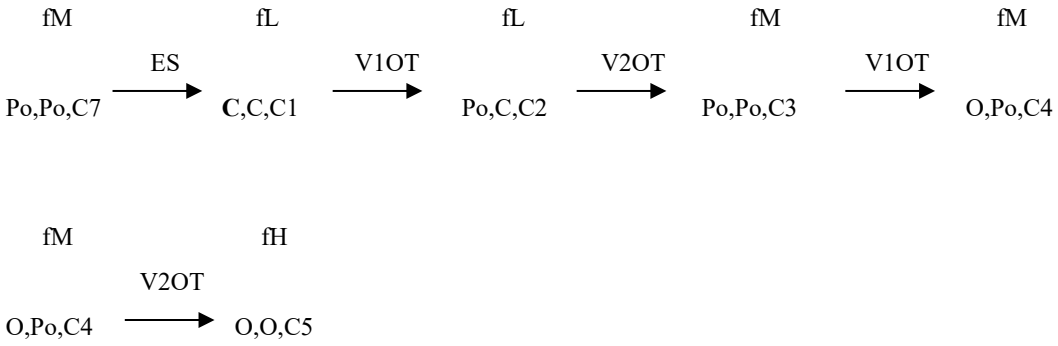


Figure 3.20: H3 System Progression

H3 is compliant with the true model future progression, so it is **kept**.

So far, it is noted that all the hypotheses can generate a future response of labels complying with the true model. This means that if we start with a nominal system in which it is detected that ES is missing in V1 which is common between V1 and V2 and follow the given controller loop , then it will be equally possible to hypothesize the missing event ES in V1 between the source state Po and any non-fault state {C,Po,O}. In other words, future observations do not resolve the ambiguity about the missing transition.

As noted in this test, the sequence of events dictated by the controller automaton made the valve V1

to be in a position dominated by V2 where the state of V2 finally decides the output label regardless of the used hypothesis.

In the next experiment, it will be assumed that the control sequence has the freedom to change its sequence in such a way V1 is the last controlled element in the sequence after V2 to see if the incompliancy could be detected or not.

3.5.3.3 Experiment C-3

The same configuration as the previous test is used here where the nominal model is built while the transition with **ES** between **Po** and C in **V1** is assumed to be mistakenly omitted. The control sequence will be modified as shown where the V2 will be commanded before V1 and sequence of events will be V2OT, V1OT, V2OT, V1OT, V1CT, V2CT, V1CT, V2CT. This is shown in Figure 3.21.

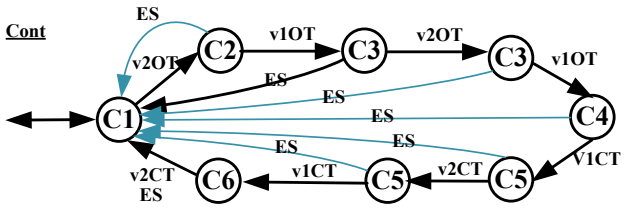


Figure 3.21: Modified Control Sequence

True model

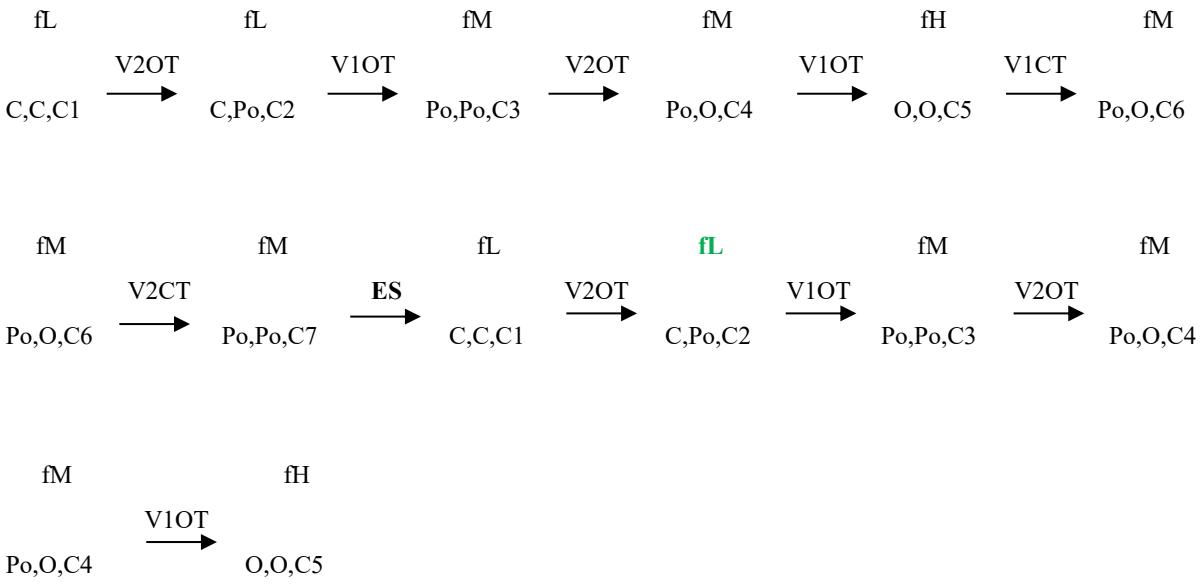


Figure 3.22: Experiment C-3 True model

H1= V1:Po $\xrightarrow{ES}$ O

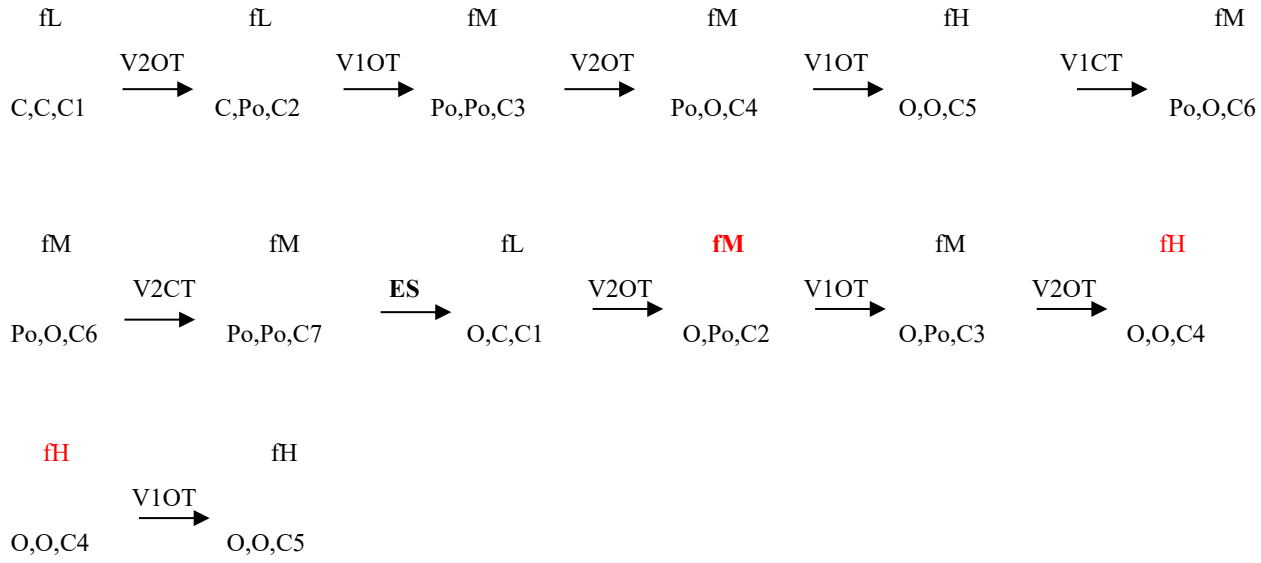


Figure 3.23: Experiment C-3 H1 Progression

H1 is incompliant with the true model future progression, as it violates the actual generated outputs after **ES**,**V2OT**. Thus, **H1** is **rejected**.

H2= V2:Po $\xrightarrow{ES}$ Po

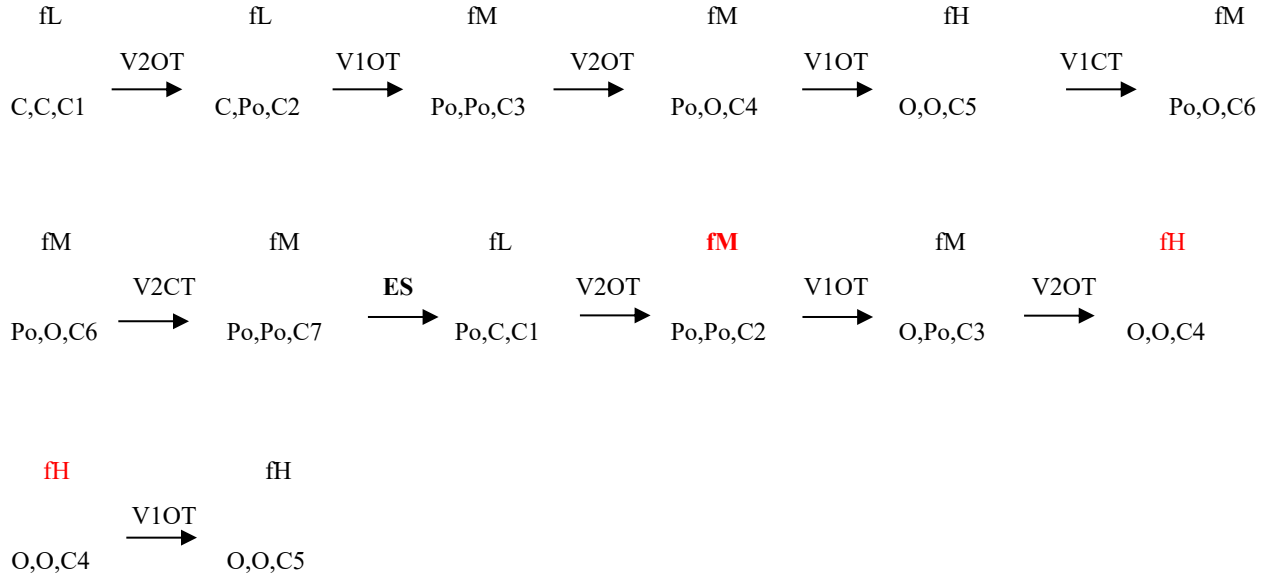


Figure 3.24: Experiment C-3 H2 Progression

H2 is incompliant with the true model future progression, as it violates the actual generated outputs after **ES**,**V2OT**. Thus, **H2** is **rejected**.

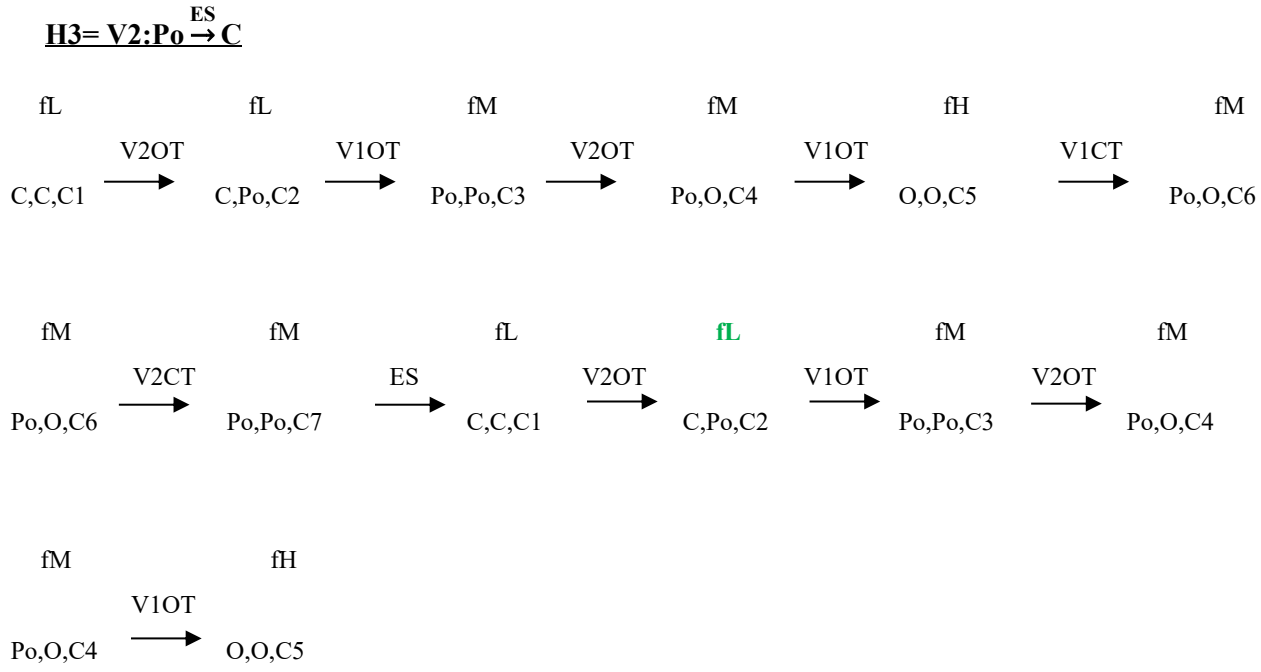


Figure 3.25: Experiment C-3 H3 Progression

H3 is compliant with the true model future progression, so it is **accepted**. As shown here, in some cases the incorrect hypotheses could be rejected if the system has some freedom of generating the sequence of events otherwise, the equal hypotheses will be considered.

3.5.3.4 Experiment C-4

This experiment has the same configuration as Experiment C-1 but the nominal model will be built with ES is mistakenly omitted in V1 between Po and C as well as in V2 between Po and C. This is to study the effect of missing this common event in multiple components in the plant. This could happen when the models of V1 and V2 come from the same database and the model error originates from the database. The control sequence will be the same as Experiment C-1 as V1OT, V2OT, V1OT, V2OT, V2CT, V1CT, V2CT, V1CT.

True model

Nominal model: Missing ES between Po and C in V1 and V2

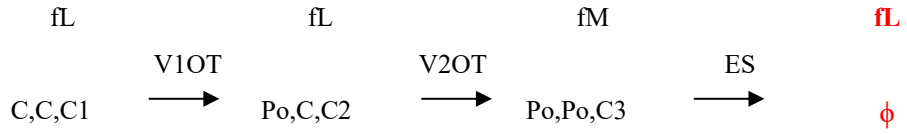


Figure 3.26 : Experiment C-4 True and Nominal Model Executions.

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? Missing Event ES

   c. ES belongs to the alphabet of V1,V2 and Cont, thus the suspected components ={Cont,V1,V2}

   d. **Last Conforming State(LCS)** in the Nominal model:(Po,Po,C3)

   e. **Actual Label of Discrepancy (ALD)** = fL.

2- **Step-2 Analysis:**

   a. Check the first suspected component **Cont**:

   - The state **C3** is the state in **Cont** when the nominal model reached the last conforming state Po,Po,**C3**.
   - At **C3**, the event **ES** is found defined.
   - Then, remove **Cont** from the suspected components.

   b. Check the next suspected component **V1**:

   - The state **Po** is the state in **V1** when the nominal model reached the last conforming state **Po**,Po,C3.
   - At **Po**, the event **ES** is found **undefined**.
   - Then, **V1** is left in the suspected components.

   c. Check the next suspected component **V2**:

   - The state **Po** is the state in **V2** when the nominal model reached the last conforming state Po,**Po**,C3.
   - At **Po**, the event **ES** is found **undefined**.

- Then, **V2** is left in the suspected components.

d. From the Last Conforming State (Po,Po,C3), **V1** and **V2 which are the suspected components** was at State **Po** in **V1** and in **Po** in **V2**.

- Thus, The **source** of missing transition in V1 = Po and in V2 = Po

Note: The hypotheses based on single missing transition in V1 or in V2 cannot explain the observations and are rejected. That is why hypotheses with two missing transitions are considered.

$V1{:}PO \xrightarrow{ES} ??$

$V2{:}PO \xrightarrow{ES} ??$

Where after the synchronous product of the components they will result in a in a label of **fL** (as the Actual label of discrepancy).

a. From the output map, list all rows that can have the Actual Label of Discrepancy (ALD)=fL

Table 3.14: Filtered Output Map to the Label fL

V1	V2	Output
C	C	fL
C	Po	fL
C	O	fL
C	SC	fL
C	SO	fL
SC	C	fL
SC	Po	fL
SC	O	fL
SC	SC	fL
SC	SO	fL
Po	C	fL
O	C	fL
SO	C	fL
Po	SC	fL
O	SC	fL
SO	SC	fL

- There are no irrelevant components and both components are suspected.
- Since the discrepancy was due to missing a transition with an observable event, the transition should not end with a faulty state that is reachable by unobservable event that were assumed that they did not occur. Thus, ES should move V1 from Po to a normal state and V2 from Po to a normal state.

Table 3.15: Candidate States

V1	V2	Output
C	C	fL
C	Po	fL
C	O	fL
Po	C	fL
O	C	fL

- From the table, we can find the combination candidates = {(C,C), (C,Po), (C,O), (Po,C),(O,C)} could be the ***candidate expected states*** for the missing ES transition. Each of these candidate combinations will make a hypothesis.

3- **Step-3: Hypothesize:**

a. Create a direct transition from the source to the candidate destination on the form: Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

H1= V1:Po $\xrightarrow{ES}$ C AND V2:Po $\xrightarrow{ES}$ C

H2= V1:Po $\xrightarrow{ES}$ C AND V2:Po $\xrightarrow{ES}$ Po

H3= V1:Po $\xrightarrow{ES}$ C AND V2:Po $\xrightarrow{ES}$ O

H4= V1:Po $\xrightarrow{ES}$ Po AND V2:Po $\xrightarrow{ES}$ C

H5= V1:Po $\xrightarrow{ES}$ O AND V2:Po $\xrightarrow{ES}$ C

4- **Step-4: Update the model**

a. Re-combine all the components based on the new hypotheses in the suspected components. **Five** hypothetical models are generated based on the three hypotheses **H1**,**H2**,**H3,H4** and **H5**.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

Thus, after ES, the future progression for the true model and the hypothetical models generated by hypotheses as follows:

True model

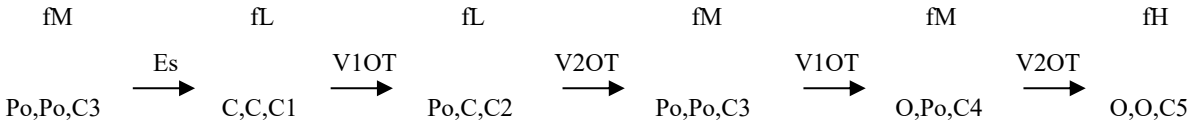


Figure 3.27: Experiment C-4 Complete System Progression

H1= V1:Po $\xrightarrow{ES}$ C AND V2:Po $\xrightarrow{ES}$ C

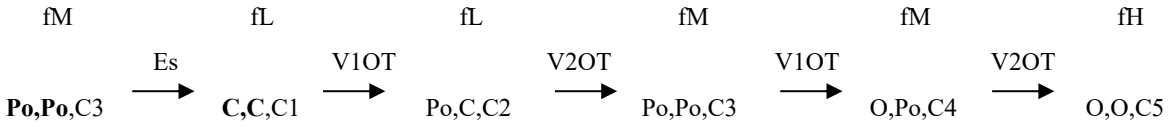


Figure 3.28: Experiment C-4 H1 Progression

H1 complies with the future progression of the true model. H1 is so far **accepted**.

H2= V1:Po $\xrightarrow{ES}$ C AND V2:Po $\xrightarrow{ES}$ Po

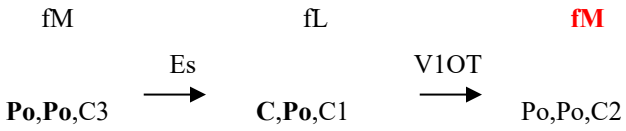


Figure 3.29: Experiment C-4 H2 Progression

H2 is incompliant with future progression of the true model. H2 is **rejected.**

H3= V1:Po $\xrightarrow{ES}$ C AND V2:Po $\xrightarrow{ES}$ O

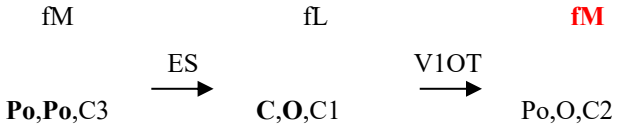


Figure 3.30: Experiment C-4 H3 Progression

H3 is incompliant with future progression of the true model. H3 is **rejected.**

H4= V1:Po $\xrightarrow{ES}$ Po AND V2:Po $\xrightarrow{ES}$ C

Figure 3.31: Experiment C-4 H4 Progression

H4 is complies with the future progression of the true model. H4 is **accepted**.

H5= V1:Po $\xrightarrow{ES}$ O AND V2:Po $\xrightarrow{ES}$ C

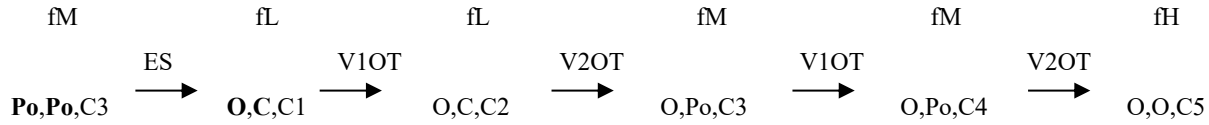


Figure 3.32: Experiment C-4 H5 Progression

H5 complies with the future progression of the true model. H5 is **accepted**. Thus, **H1**, **H4** and **H5** are accepted hypotheses.

3.5.3.5 Experiment C-5

Let us assume that in Experiment-C4, the controller followed a different sequence given below:

V2OT, V1OT, V2OT, V1OT, V1CT, V2CT, V1CT, V2CT as shown in Figure 3.33. In this case we will see that H1 will be kept and H4 and H5 will be rejected.

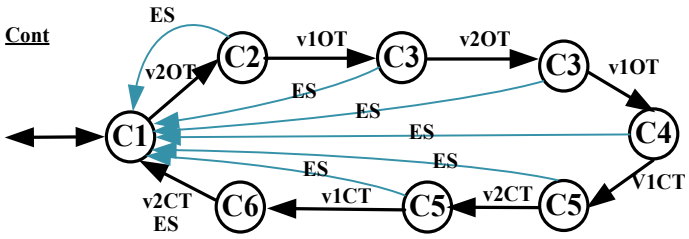


Figure 3.33: Modified Control Sequence

True model

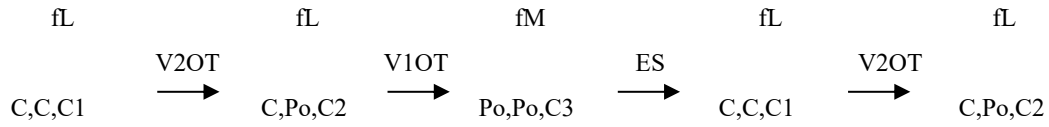


Figure 3.34: True model Progression

H1= V1:Po $\xrightarrow{ES}$ C AND V2:Po $\xrightarrow{ES}$ C

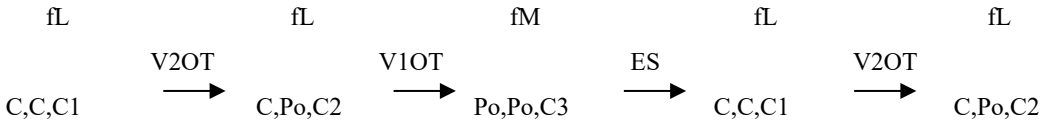


Figure 3.35: Experiment C-5 H1 Progression

H1 complies with the future progression of the true model. H1 is **kept**.

H4= V1:Po $\xrightarrow{ES}$ Po ANDV2:Po $\xrightarrow{ES}$ C

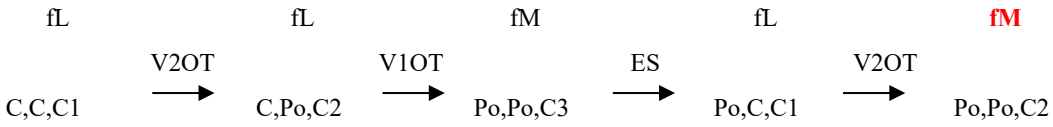


Figure 3.36: Experiment C-5 H4 Progression

H4 is incompliant with future progression of the true model. H4 is **rejected.**

H5= V1:Po $\xrightarrow{ES}$ O AND V2:Po $\xrightarrow{ES}$ C

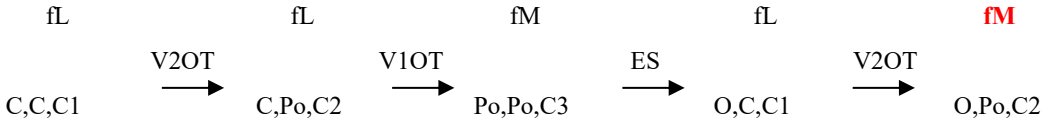


Figure 3.37: Experiment C-5 H5 Progression

H5 is incompliant with future progression of the true model. H5 is **rejected.**

3.5.4 Derived Rules for the Learning Algorithm

The following rules are derived from the previous experiments and will be used in constructing the algorithm that finds the missing transitions in the case of observable events. The rules are named as ObRule-n, where the prefix Ob stands for Observable events while n is the rule number.

ObRule-1: At the Last Conforming State (LCS), every component that has the event of discrepancy in its event set yet undefined at LCS is a suspected component.

ObRule-2: The Last Conforming State (LCS) gives the source of the missing transitions in suspected

components.

ObRule-3: The event of discrepancy shall move the nominal model to a destination state with the same output as the **ALD**. The possible destinations are determined using the output map. The selection could have more than one candidate destination state. Thus, the candidate destination states of the nominal model give the destination of missing transitions.

ObRule-4: The parsimonious logic is used to explain the discrepancy using the least number of transitions, by filtering the possible destinations to the candidate destinations. This will be carried out in **ObRule-5**, **6** and **7**.

ObRule-5: The irrelevant components - that do not have the discrepancy event in their alphabets - shall not change their state.

ObRule-6: The complete suspected component – having the discrepancy event in its alphabet but defined at LCS - shall progress with the common event as defined in their automata.

ObRule-7: The Observable event shall move a suspected component to a normal candidate state and NOT to a faulty state.

ObRule-8: A hypothesis is defined based on a possible missing transition from a source to a destination with the event of discrepancy.

ObRule-9: The number of hypothetical models equals the number of hypotheses of the possible missing transitions missing transitions.

ObRule-10: In terms of the future progression of the system, the compliant models are kept, and the incompliant models are rejected.

ObRule-11: Equal hypotheses are reduced to one hypothesis per transition if they pertain to the same component so that they could be tested separately. If the Equal hypotheses pertain to different components, they can be merged in a combined hypothesis.

3.6 Missing Transitions with Unobservable Events

This section we will study the case of missing transitions involving unobservable events. The reason for missing these transitions in the model could be human mistakes, lack of knowledge or the simplification of the system. A set of experiments will be operated as done in the previous section in order to develop some rules to generate hypotheses for the missing transitions.

3.6.1 Assumptions of the Experiments

1. The components, e.g., valve or pump are modeled with the correct number of states and are not missing any states.
2. Only some transitions with unobservable events are missing in the automata of the components.
3. The missing transitions have unobservable events that are known in the alphabet, and they are not new events to be added to the alphabet.
4. The output map is known and accurate therefore it is the same for the true and nominal model.
5. The fault modes are permanent where if a component fails to a certain fault mode, it remains indefinitely. This is by following the same diagnosability assumption in [20].

We start with a simple system then we will add more components. The plant is comprised of two valves V1 and V2. The automata of the valves are shown in Figure 3.38.

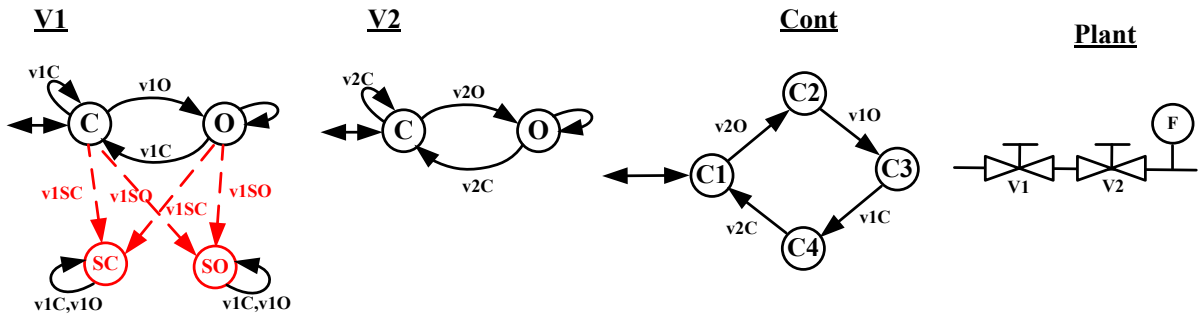


Figure 3.38: Demonstration System for Unobservable Events

The first valve automaton **V1** is a two-state valve of open and closed positions modeled by automaton **V1**. The states are:

C: Closed. **O**: Open. **SC**: Stuck Closed. **SO**: Stuck Open.

The events symbols are defined as follows:

V1C: Close command. **V1O**: Open Command.

V1SC: Stuck Closed fault. **V1SO**: Stuck Open fault.

The second valve automaton **V2** is a two-position valve of open and closed but it was considered reliable having no faults. This fault-free assumption is to simplify the discussion.

The states are:

C: Closed. **O**: Open.

The events symbols are defined as follows:

V2C: Close command. **V2O**: Open Command.

The controller **Cont** issues the control sequence for this system, and is modeled by automaton **Cont.** The control sequence will generate the sequence of events V2O,V1O,V1C,V2C and so on.

The output labels given by the flowmeter F are:

fL: Low flow **fH**: High flow

Note: The experiments will be numbered as Experiment-Un, where U stands for unobservable events while n is the experiment number.

3.6.1.1 Experiment U-1

In this experiment, the nominal model is built by deleting the transition between C and SO with the unobservable event V1SO as indicated with the crossed transition in Figure 3.39. Here it has been assumed mistakenly in the nominal model that stuck-open failure does not happen in the valve's closed position.

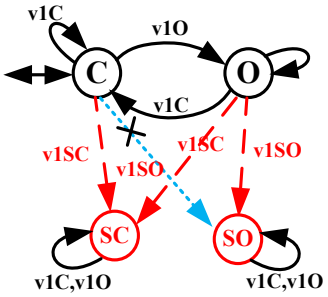
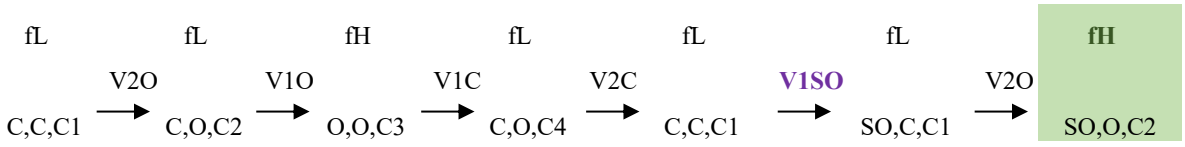


Figure 3.39: V1 After Deleting V1SO Between C and SO

True model:



Nominal Model: Missing V1SO between C, SO in V1

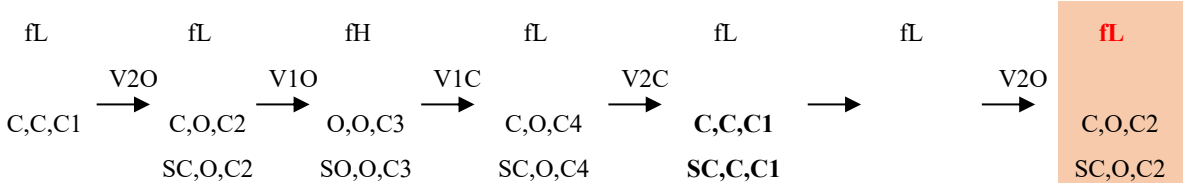


Figure 3.40: Experiment U-1 True and Nominal Models

Firstly, because of having unobservable events in the model, some states could be expected to have more than one possible state. For example, in the state chart of the nominal model and starting from C,C,C1 when V2O occurs, the model moves to C,O,C2 but it is possible also that V1 has stuck-closed so SC,O,C2 is another possible state.

As shown in the state chart in Figure 3.40, the nominal model complies with the true model until it reaches the possible states {C,C,C1, SC,C,C1} where the output in both models is fL. Then, the unobservable event V1SO occurred in the plant making V1 fail to open. This fault event is unobservable by measurements and its transition is not included in the nominal model at the last states set {C,C,C1, SC,C,C1}. Once a new observable event V2O occurs, the nominal model expects mistakenly the output fL because it considers V1 closed or stuck-closed. This is while the real system represented by the true model generated a different actual output fH because the V1 failed to stuck-open then V2 is commanded to open. This mismatch between the output expected by the nominal model (fL) and the real system (fH) is the sign of a discrepancy caused by missing a transition with unobservable event.

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? **Wrong expected output after observable event.**

   c. Expected State(ES)= (C,O,C2) or (SC,O,C2)

   d. Last Conforming State(LCS) in the Nominal model: {(C,C,C1), (SC,C,C1)}

   e. Actual Label of Discrepancy (ALD) = fH.

2- **Step-2 Analysis:**

To determine the suspected components, the output map will be used to select the states of the system that have the same output as the Actual Label of Discrepancy (ALD) = fH. These states are suspected to be a destination of the missing transition and listed in a table that will be referred to as the **Suspected States** table or the **SS-table**.

SS-Table:

Table 3.16: Experiment U-1 SS-Table

V1	V2	Label
O	O	fH
SO	O	fH

   a. Reject the rows with all normal states.

Since the current experiment discrepancy is in the output label and not the generated observable

events, the cause of the discrepancy must be an unobservable event, in this case, a failure event and the destination state should not be a normal state. In other words, the missing transition should end in a state reachable by any of the unobservable events. Therefore, in the SS-table above, the state (O,O) is ruled out. By comparing the last reached or Expected States (ES) = (C,O,C2) and (SC,O,C2) in the nominal model with the actual state SO,O,C2 in the true model, we can see that V2 is correctly expected to be at O while V1 was wrongly expected to be at C or SC rather than SO. This means generally, in the expected states ES we can see that there are correctly expected states belonging to some components that will be called **correctly expected components.** Similarly, in the ES, there are wrongly expected states belonging to some components that will be called **wrongly expected components.** The wrongly expected components are the suspected components. Since as mentioned before, during the experiments, we use a known true model but in practice, the real system is modeled with the best of knowledge as the nominal model while the true model is unknown. Thus, it is required to find a way to know the wrongly expected components. This will be done as follows:

b. From the SS-table, select the states that have some components conforming with **Expected State** (ES)=(C,O,C2) or (SC,O,C2). In other words, select the states in which the components have similar values as the values of components in the expected state ES. This will be explained in detail in the following step.

For now, we will omit the controller as it is assumed to be programmatically known and introduced by the designer. The focus will be to correct the physical components. Thus, we can consider ES = {(C,O), (SC,O)}.

Considering the first ES = (C,O):

- To find conforming states in the SS-table, let us hold V2 constant at **O**, and V1 could be variable. So, we anticipate finding (SC,**O**) or (SO,**O**). But the anticipated (SC,**O**) is not in the SS-table because it does not make the output label to fH as the **ALD,** so it is rejected. On the other hand, (SO,**O**) exists in the SS-table, so it is a candidate.
- Hold V1 constant at **C** and V2 could be variable. So, we anticipate finding (**C**,SO) or (**C**,SC) but both do not exist in SS-table. This is because V2 in this example does not have neither SC nor SO.

ES	V1 Variable and V2 Constant	V1 Constant , V2 Variable
(C,O)	~~*(SC,**O**)*~~ *or (SO,**O**)*	~~*(**C**,SO)*~~ *or* ~~*(**C**,SC)*~~ Both do not exist

Considering the second ES = (SC,O):

- Hold V2 constant at **O**, and V1 could be variable. So, we expect to find (SC,**O**) or (SO,**O**). But

(SC,O) is not in the SS-table because it does not make the output label to fH as the **ALD,** so it is rejected. On the other hand (SO,**O**) exists in the SS-table, but here it cannot be considered because the assumption of permanent fault holds such that when V1 is in SC it remains and does not move to SO. However (SO,O) has been already candidate previously.

- Hold V1 constant at SC and V2 could be variable. So, we expect to find (**SC**,SO) or (**SC**,SC) but both do not exist. This is because V2 in this example does not have neither SC nor SO.

ES	V1 Variable and V2 Constant	V1 Constant , V2 Variable
(SC,O)	~~*(SC,**O**)* or *(SO,**O**)*~~	~~*(**SC**,SO)* or *(**SC**,SC)*~~ Both do not exist

c. (SO,**O**) is found as a *candidate expected state* for the first ES = {(C,**O**), where we can see that V2 is correctly expected to be O, and V1 should be SO.

d. Since the value of V2 is **correctly expected** and the **wrongly expected** is **V1**, thus **suspected components**={V1}.

e. Source of missing transition: in the Last Conforming State (LCS) = (**C**,C,C1), where V1 is at C, then the source of missing transition in V1 is C.

f. Transition Destination Candidates: from the found records (the candidate expected state) of the SS-table, V1 should be at SO, thus destination candidate in V1 is SO.

3- Step-3: Hypothesize:

a. Create a direct transition from the source to the candidate destination on the form:

b. Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

c. In V1, Transition Destination Candidates = SO is reachable through V1SO.

d. Thus, in V1 Hypothesis

$$H1 = V1{:}C \xrightarrow{V1SO} SO$$

4- Step-4: Update the model

a. Re-combine all the components based on the new hypotheses in the suspected components.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

c. In this experiment, we have only one hypothesis that is retrieved exactly as the complete system so logically the updated nominal model will comply with the true model and no need to check the future progression in this case.

3.6.1.2 Experiment U-2

In this experiment, the plant will be comprised of two similar valves as the previous experiment but in parallel. In addition, the cross failures (from O to SC and C to SO) will not be considered. The nominal model will be built by deleting the unobservable event V1SO between O and SO in V1. The controller will have the sequence of events V1O,V2O, V2C,V1C as shown in Figure 3.41.

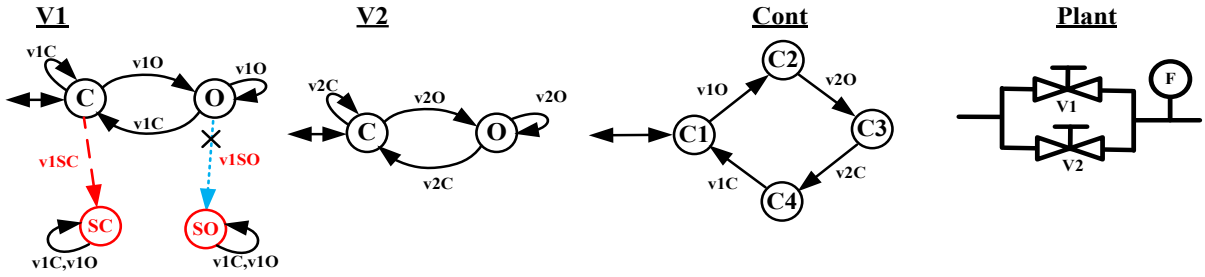


Figure 3.41: Experiment U-2 System

True model:

fL		fH		fH		fH		fH		fH		fH
1	V1O		V2O		**V1SO**		V2C		V1C		V1O	
C,C,C1	→	O,C,C2	→	O,O,C3	→	SO,O,C3	→	SO,C,C4	→	**SO**,C,C1	→	SO,C,C2

Nominal Model: missing V1SO in V1 between O and SO

fL		fH		fH		fH		fH		fL
1	V1O		V2O				V2C		V1C	
C,C,C1	→	O,C,C2	→	O,O,C3	→			**O,C,C4**	→	**C,C,C1** **SC,C,C1**

Figure 3.42: Experiment U-2 True and Nominal Models

1- **Step-1 Information Collection**

- **a.** Is there a discrepancy detected? Yes
- **b.** Type of Discrepancy? **Wrong expected output after an observable event.**
- **c.** Expected State(ES)= (C,C,C1) or (SC,C,C1).
- **d.** Last Conforming State(LCS) in the Nominal model: (O,C,C4).
- **e.** Actual Label of Discrepancy (ALD) = fH.

2- **Step-2 Analysis:**

Suspected States table or the (SS-table) for the states in the model that can have the same output as Actual Label of Discrepancy ALD= fH.

SS-Table:

Table 3.17: Experiment U-2 SS-Table

V1	V2	Output
O	O	fH
C	O	fH
O	C	fH
SO	O	fH
SC	O	fH
SO	C	fH

a. Reject the rows with all normal states (looking for faulty states or states reachable by unobservable events).

Table 3.18: Experiment U-2 Filtered SS-Table

V1	V2	Output
SO	O	fH
SC	O	fH
SO	C	fH

b. From the filtered SS-table, select the states that have some components conforming with Expected States (ES) = {(C,C,C1), (SC,C,C1)}. By neglecting the controller component Cont, then ES becomes = {(C,C), (SC,C)}

For (C,C): V2 is found to be C in one record in the table (V1=SO, V2=C), thus (SO,C) is the corresponding *candidate expected state.*

For (SC,C): V1 is found to be SC in one record in the table (V1=SC, V2=O), thus (SC,O) is the corresponding *candidate expected state.*

Also, V2 is found to be C in one record in the table (V1=SO, V2=C), thus (SO,C) is the corresponding *candidate expected state.*

The summary

Table 3.19 below summarizes the conclusion of sources and destinations of the missing transitions and is explain as follows:

Table 3.19: Experiment U-2 Source and Destination Summary Table

LCS	ES	Candidate Expected States	Suspected Component	Source	Destination
O,C	**C**,C	(**SO**,C)	V1	O	SO
	SC,C	(SC,O)	V1	O	SC
			V2	C	O
		(SO,C)	V1	O	SO

c. For the first ES = (C,C)

i. The candidate expected state (SO,**C**) when compared to ES = (C,**C**), we can find that V1 is wrongly expected, thus suspected components={V1}.

ii. Source of missing transition: in the Last Conforming State (LCS)=(O,C), where V1 is at O, then the source of missing transition in V1 is O.

iii. Transition Destination Candidate: from the found records (the candidate expected state) of the SS-table, V1 should be at SO, thus destination candidate in V1 is SO.

d. For the second ES = (SC,C)

i. The first candidate expected state (SC,O) compared to ES = (SC,C), we find that V1 and V2 are wrongly expected, thus suspected components={V1,V2}.

- Source of missing transition: in the Last Conforming State (LCS)=(O,C), where V1 is at O, V2 is at C.

  Then the source of the first missing transition in V1 is O.

  The source of the second missing transition in V2 is C.

- Transition Destination Candidates:

  For the second candidate expected state (SC,O),

  V1 should be at SC, thus the destination of first missing transition is SC.

  V2 should be at O, thus the destination of first missing transition is O.

ii. The second candidate expected state (SO,**C**) when compared to ES = (SC,**C**), we can find that V1 is wrongly expected, thus suspected components={V1}.

- Source of missing transition: in the Last Conforming State (LCS)=(O,C), where V1 is at O, then the source of missing transition in V1 is O.
- Transition Destination Candidate: from the found records (the candidate expected state) of the SS-table, V1 should be at SO, thus destination candidate in V1 is SO.

3- Step-3: Hypothesize:

a. Create a direct transition from the source to the candidate destination on the form:

b. Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

- For first expected state ES = (C,C)

H1= V1:O $\xrightarrow{V1SO}$ SO as SO is a state reachable by V1SO.

- For the second expected state ES = (SC,C).

  H2= V1:O $\xrightarrow{V1SC}$ SC as SC is reachable by V1SC, but this hypothesis is **rejected** by logic because this transition is not included as mentioned in the system setup.

  H3= V1:C $\xrightarrow{Unobservable\ event??}$ O This hypothesis is **rejected** by logic since O is reachable by observable events.

  H4= V1:O $\xrightarrow{V1SO}$ SO as SO is a state reachable by V1SO, but this hypothesis is rejected because it is repeated in H1.

4- **Step-4: Update the model**

   a. Re-combine all the components based on the new hypotheses in the suspected components.

   b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

   c. In this experiment, we have only one hypothesis that is retrieved exactly as the true model, so logically the updated nominal model will comply with the true model and no need to check the future progression in this case.

3.6.1.3 Experiment U-3

In this experiment, the plant will be comprised of the same two valves as the previous experiment in parallel. The nominal model is built by removing the unobservable event V1SO between C and SO in V1 as shown in Figure 3.43.

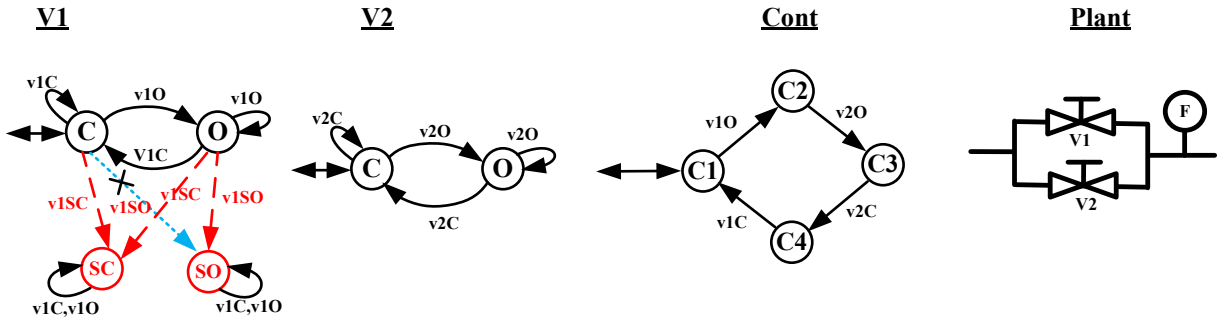
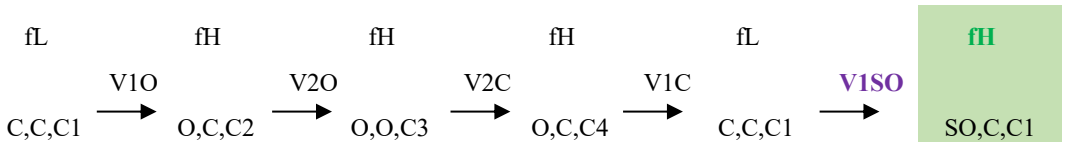


Figure 3.43: Experiment U-3 System

True model:



Nominal Model: Missing V1SO between C,SO in V1

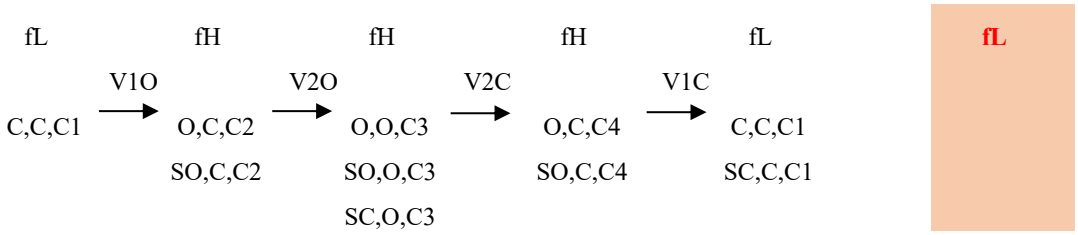


Figure 3.44: Experiment U-3 True and Nominal Models

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? **Wrong unexpected output.**

   Note: Here the nominal model was matching the system until it reached the state (C,C,C1) with output=fL and suddenly the real system generated a new different output **fH** while the currently expected output is **fL**. This sudden change happened due to an unobservable event that could not be explained by the nominal model.

   c. Expected State(ES)=(C,C,C1)

   d. Last Conforming State(LCS) in the Nominal model: (C,C,C1)

   Note: in this type of discrepancy (wrong unexpected output), **ES** equals **LCS**.

   e. Actual Label of Discrepancy (ALD) = fH.

2- **Step-2 Analysis:**

Suspected States table or the (SS-table) for the states in the model that can have the same output as Actual Label of Discrepancy ALD= fH.

SS-table:

Table 3.20: Experiment U-3 SS-Table

V1	V2	Output
O	O	fH
C	O	fH
O	C	fH
SO	O	fH
SC	O	fH
SO	C	fH

a. Reject the rows with all normal states (looking for states reachable by unobservable events which are faulty states in this experiment).

Table 3.21: Experiment U-3 Filtered SS-Table

V1	V2	Output
SO	O	fH
SC	O	fH
SO	C	fH

e. For the first ES = (C,C,C1)

iv. From the output map table, select the states that have some components conforming with Expected State (ES)= (C,C,C1).

v. V2 is found to be C in one record in the table (V1=SO, V2=C)
Thus, (V1=SO, V2=C) is the *candidate expected state.*

vi. The wrongly expected component is V1, thus suspected components={V1}

vii. Source of missing transition: in the expected state ES =(C,C,C1), where V1 is at C, then the source of missing transition in V1 is C.

viii. Transition Destination Candidates: from the found records (the candidate expected state) of the SS-table, V1 should be at SO, thus destination candidate in V1 is SO.

f. For the second ES = (SC,C,C1)

ix. From the output map table, select the states that have some components conforming with Expected State (ES)= (SC,C,C1).

x. V2 is found to be C in one record in the table (V1=SO, V2=C)
Thus, (V1=SO, V2=C) is the *candidate expected state.*

xi. The wrongly expected component is V1, thus suspected components={V1}

xii. Source of missing transition: in the expected state ES =(SC,C,C1), where V1 is at SC, then the source of missing transition in V1 is SC.

xiii. Transition Destination Candidates: from the found records (the candidate expected state) of the SS-table, V1 should be at SO, thus destination candidate in V1 is SO.

xiv. V1 is found to be SC in one record in the table (V1=SC,V2=O)
Thus, (V1=SC, V2=O) is the *candidate expected state.*

xv. The wrongly expected component is V2, thus suspected components={V2}

xvi. Source of missing transition: in the expected state ES =(SC,C,C1), where V2 is at C, then the source of missing transition in V2 is C.

xvii. Transition Destination Candidates: from the found records (the candidate expected state) of the SS-table, V2 should be at O, thus destination candidate in V2 is O.

Table 3.22: Experiment U-3 Source and Destination Summary Table

LCS	**ES**	**Candidate Expected States**	**Suspected Component**	**Source**	**Destination**
C,C	**C**,C	(**SO**,C)	V1	C	SO
SC,C	SC,C	(SO,C)	V1	SC	SO
		(SC,O)	V2	C	O

3- Step-3: Hypothesize:

a. Create a direct transition from the source to the candidate destination on the form:

b. Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

- For first expected state ES = (C,C)

  H1= V1:C $\xrightarrow{\text{V1SO}}$ SO as SO is a state reachable by V1SO.

- For the second expected state ES = (SC,C).

  H2= V1:SC $\xrightarrow{\text{V1SO}}$ SO as SO is reachable by V1SO, but this hypothesis is **rejected** based on the assumption that the fault modes are permanent.

  H3= V2:C $\xrightarrow{\textit{Unobservable event??}}$ O This hypothesis is **rejected** by logic since O is reachable by observable events.

4- Step-4: Update the model

a. Re-combine all the components based on the new hypotheses in the suspected components.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

c. In this experiment, we have only one hypothesis that is retrieved exactly as the complete system so logically the updated nominal model will comply with the true model.

3.6.1.4 Experiment U-4

In this experiment, the plant is comprised of the same two valves as the previous experiment. The nominal model is built by removing the unobservable event V1SC between C and SC in V1 as shown in Figure 3.45.

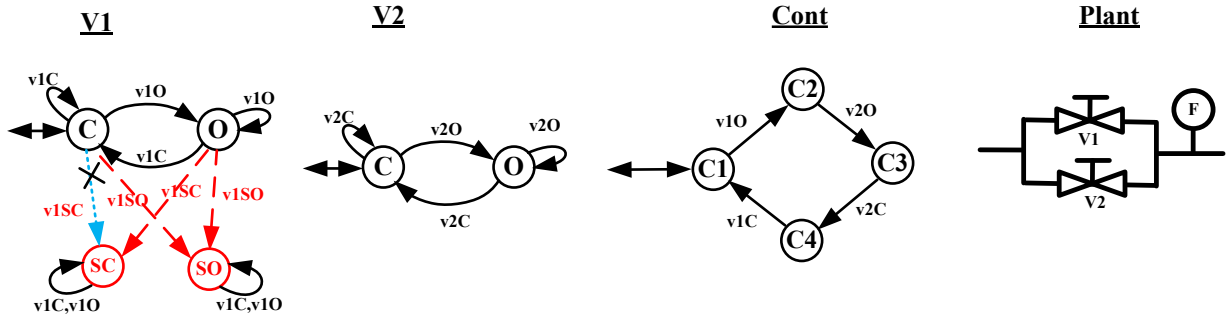
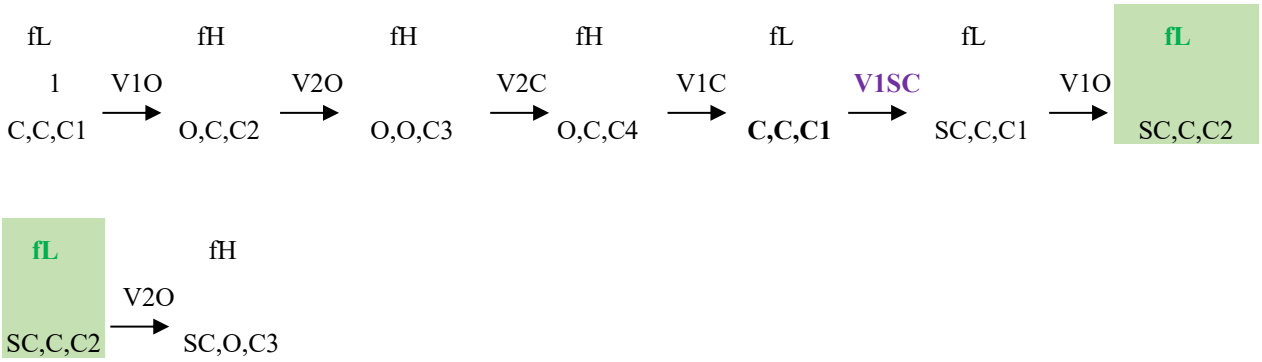


Figure 3.45: Experiment U-4 System

True model:



Nominal model: Missing V1SC between C and SC in V1

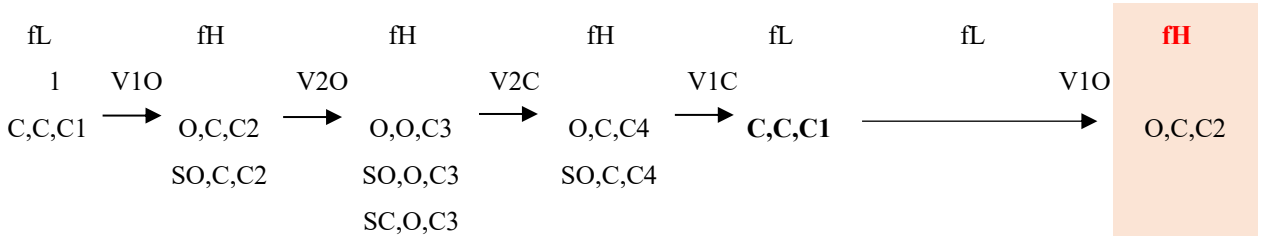


Figure 3.46: Experiment U4 True and Nominal Models

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? **Wrong expected output after an observable event.**

   c. Expected State(ES)=(**O,C,C2**)

   d. Last Conforming State(LCS) in the Nominal model: (**C,C,C1**)

   e. Actual Label of Discrepancy (ALD) = **fL**.

2- **Step-2 Analysis:**

Suspected States table or the (SS-table) for the states in the model that can have the same output as Actual Label of Discrepancy **ALD** = **fL**.

SS-table:

Table 3.23: Experiment U-4 SS-Table

V1	V2	Output
C	C	fL
SC	C	fL

   a. Reject the rows with all normal states (looking for faulty states).

Table 3.24: Experiment U-4 Filtered SS-Table

V1	V2	Output
SC	C	fL

   b. From the output map table, select the states that have some components conforming with Expected State (ES) = (C,C,C1).

   c. V2 is found to be C in one record in the table (V1=SC, V2=C)
   Thus, (V1=SC, V2=C) is the *candidate expected state*.

   d. The wrongly expected component is V1, thus suspected components={V1}.

   e. Source of missing transition: in the Last Conforming State(LCS) = (C,C,C1), where V1 is at C, then the source of missing transition in V1 is C.

   f. Transition Destination Candidates: from the found records (the candidate expected state) of the SS-table, V1 should be at SC, thus destination candidate in V1 is SC.

3- **Step-3: Hypothesize:**

   a. Create a direct transition from the source to the candidate destination on the form:

   b. Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

   c. In V1, Transition Destination Candidate = **SC** is reachable through **V1SC**.

d. Thus, in V1 Hypothesis

$H1 = V1{:}C \xrightarrow{V1SC} SC$

e. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

4- **Step-4: Update the model**

a. Re-combine all the components based on the new hypotheses in the suspected components.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

c. In this experiment, we have only one hypothesis that is retrieved exactly as the complete system so logically the updated nominal model will comply with the true model.

3.6.1.5 Experiment U-5

In this experiment, the plant will be comprised of the same two valves as the previous experiment. The nominal model is built by deleting the unobservable event V1SC between O and SC in V1 as shown in Figure 3.47.

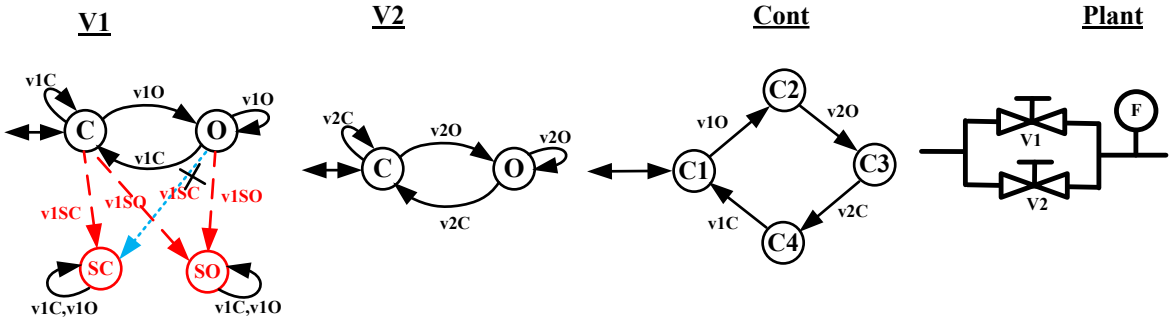
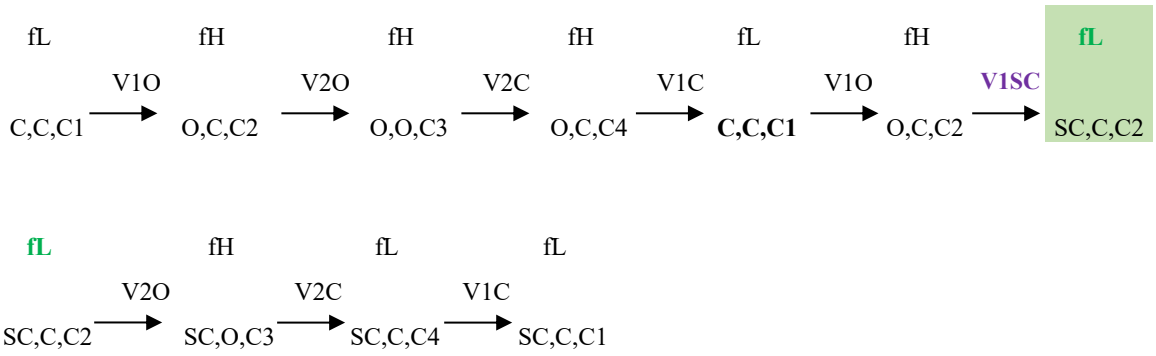


Figure 3.47: Experiment U-5 System

True model:



Nominal model:

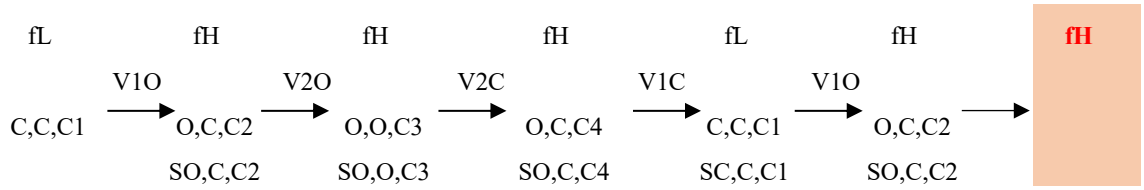


Figure 3.48: Experiment U-5 True and Nominal Models

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? **Wrong unexpected output.**

   Note: Here the nominal model was matching the system until it reached the possible states {(O,C,C2), (SO,C,C2)}with output = fH and suddenly the real system generated a new different output fL while the currently expected output is fH. This sudden change happened due to an unobservable event that could not be explained by the nominal model.

   c. Expected State (ES)={(O,C,C2), (SO,C,C2)}. By neglecting the controller component "Cont", then ES becomes = {(O,C), (SO,C)}.

   d. Last Conforming State (LCS) in the nominal model = {(O,C,C2), (SO,C,C2)}. By neglecting the controller component "Cont", then LCS becomes = {(O,C), (SO,C)}.

   Note: in this type of discrepancy (wrong unexpected output), ES equals LCS.

   e. Actual Label of Discrepancy (ALD) = fL.

2- **Step-2 Analysis:**

Suspected States table or the (SS-table) for the states in the model that can have the same output as Actual Label of Discrepancy ALD = fL.

SS-Table:

Table 3.25: Experiment U-5 SS-Table

V1	V2	Output
C	C	fL
SC	C	fL

a. Reject the rows with all normal states (looking for states reachable by unobservable events which are the faulty states in this experiment).

Table 3.26: Experiment U-5 Filtered SS-Table

V1	V2	Output
SC	C	fL

b. From the SS-table, select the states that have some components conforming with Expected State (ES)= {(O,C), (SO,C)}.

- For the first ES = (O,C): V2 is found to be C in one record in the table (V1=SC, V2=C), thus (SC,C) is corresponding *candidate expected state.*
- For the second ES = (SO,C): V2 is found to be C in one record in the table (V1=SC, V2=C), thus, (SC,C) is the corresponding *candidate expected state.*

The summary Table 3.27 below summarizes the conclusion of sources and destinations of the missing transitions and is explain as follows:

Table 3.27: Experiment U-5 Source and Destination Summary Table

LCS	ES	Candidate Expected States	Suspected Component	Source	Destination
O,C	O,C	SC,C	V1	O	SC
SO,C	SO,C	SC,C	V1	SO	SC

g. For the first ES = (O,C)

i. The candidate expected state (SC,**C**) when compared to ES = (O,**C**), we can find that V1 is wrongly expected, thus suspected components={V1}.

ii. Source of missing transition: in the Last Conforming State (LCS)=(O,C), where V1 is at O, then the source of missing transition in V1 is O.

iii. Transition Destination Candidate: from the found records (the candidate expected state) of the SS-table, V1 should be at SC, thus destination candidate in V1 is SC.

h. For the second ES = (SO,C)

i. The candidate expected state (SC,**C**) when compared to ES = (SO,**C**), we can find that V1 is wrongly expected, thus suspected components={V1}.

ii. Source of missing transition: in the Last Conforming State (LCS)=(SO,C), where V1 is at SO, then the source of missing transition in V1 is SO.

iii. Transition Destination Candidate: from the found records (the candidate expected state) of the SS-table, V1 should be at SC, thus destination candidate in V1 is SC.

3- **Step-3: Hypothesize:**

a. Create a direct transition from the source to the candidate destination on the form:

b. Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

- For the first expected state ES = (O,C)

  H1= V1:O $\xrightarrow{\text{V1SC}}$ SC as SC is a state reachable by V1SC.

- For the second expected state ES = (SO,C)

  H2= V1:SO $\xrightarrow{\text{V1SC}}$ SC as SC is a state reachable by V1SC. But this hypothesis is **rejected** by logic because it contradicts with the assumption of having permanent fault mode, where if a certain component drops in a fault mode, it remains indefinitely, as well as it does not move to another fault mode.

4- **Step-4: Update the model**

a. Re-combine all the components based on the new hypotheses in the suspected components.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

c. In this experiment, we have only one remaining hypothesis that is retrieved exactly as the complete system so logically the updated nominal model will comply with the true model.

3.6.1.6 Experiment U-6

This experiment will repeat experiment U-1 but with changing the control sequence V1O,V2O,V2C,V1O as shown in Figure 3.49. The nominal model is built by deleting V1SO between C and SO in V1 as shown in Figure 3.49.

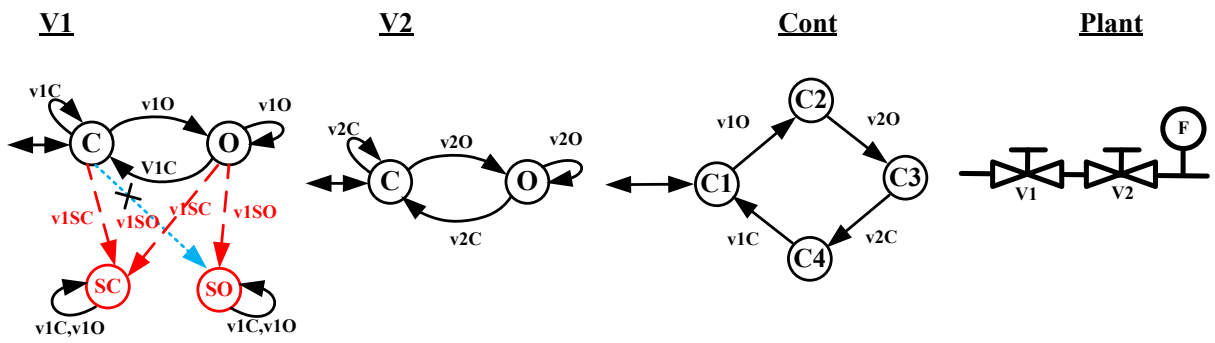


Figure 3.49: Experiment U-6 System

True model:

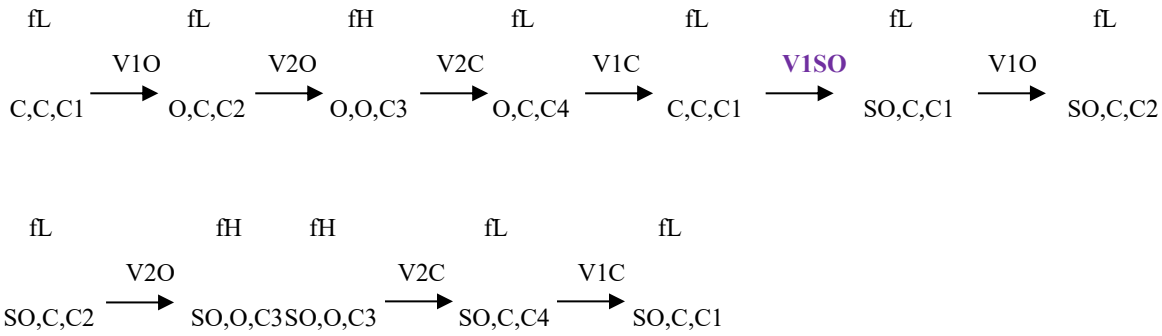


Figure 3.50: Experiment U-6 True Model

Nominal Model: **Missing V1SO between C and SO in V1**

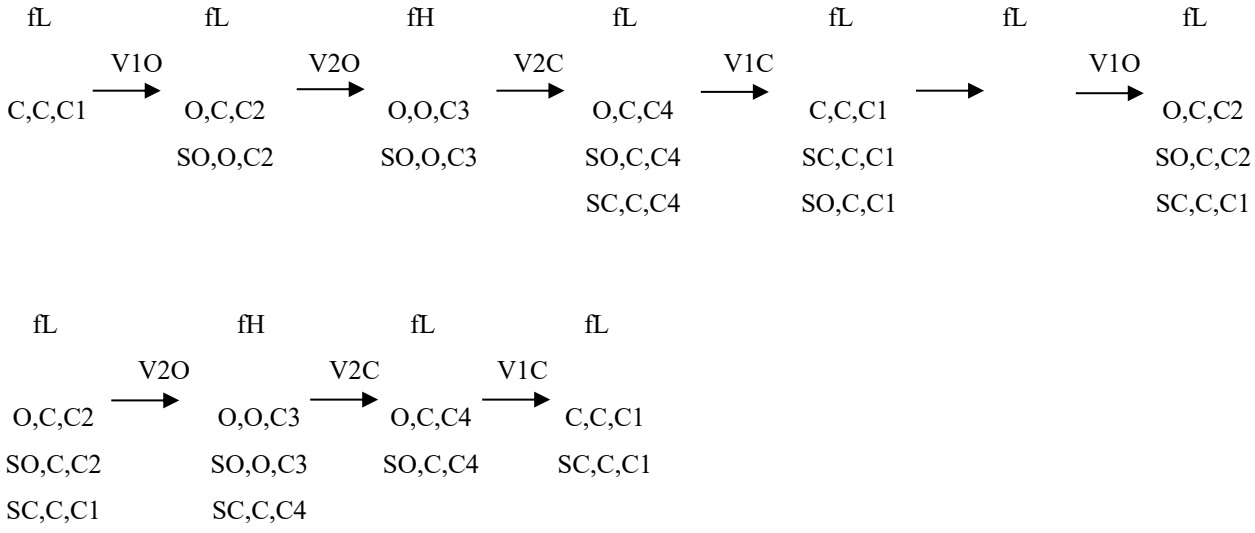


Figure 3.51: Experiment U-6 Nominal Models

It could be noted that in this experiment, the nominal model is complying with the real system which is represented by the true model in terms of following the same sequence of events and generating the same output labels. This is although the nominal model is built with a missing transition in a component compared to the true model. The positioning of the component V1 in the system with the controller along with the sequence of events applied to the system made the unobservable fault event V1SO to be undiagnosable. The principle of diagnosability was tackled in the literature as in [20] which determines the conditions upon which a diagnoser will be able to diagnose a fault caused by an unobservable event.

The diagnosability could be shown in the following example if we compared the behavior of the system (**without** having the fault V1SO) in one side. On the other side, the true model **with** the fault V1SO along with the nominal model missing the transition of V1SO in V1.

True model *without* V1SO:

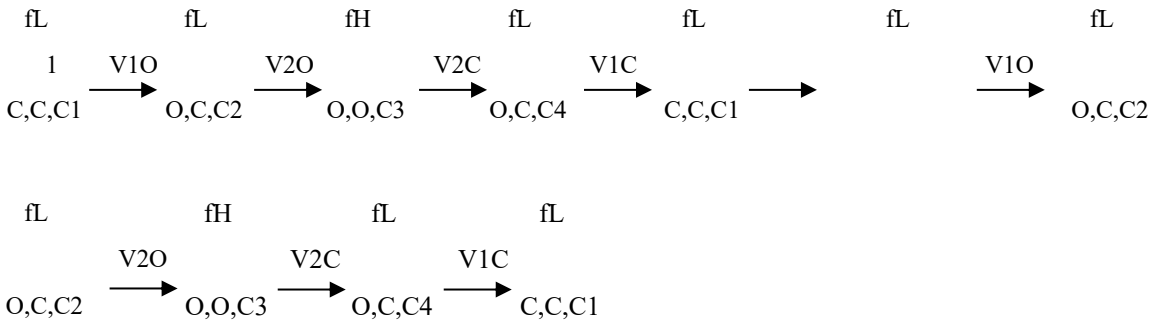


Figure 3.52: Experiment U-6: True model without V1SO

True model *with* V1SO:

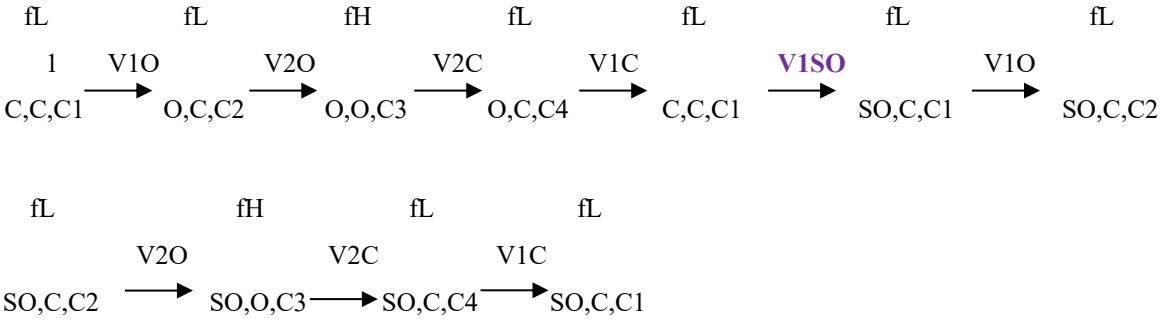


Figure 3.53: Experiment U-6: True model with V1SO

Nominal Model: Missing V1SO between C and SO in V1

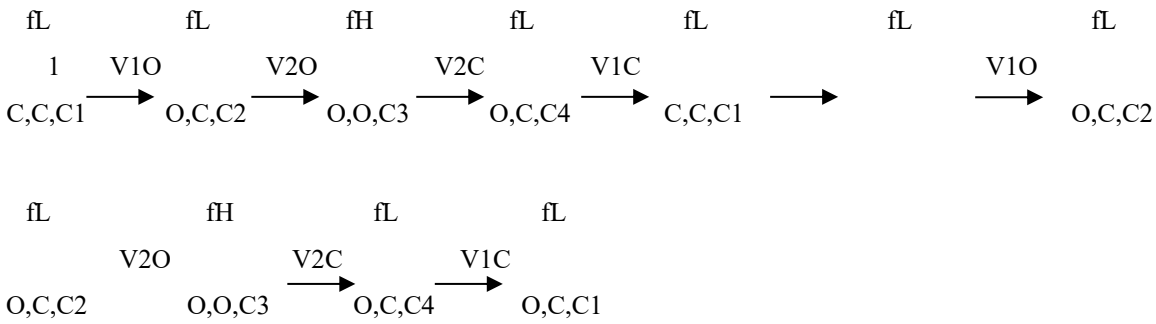


Figure 3.54: Experiment U-6 Nominal Model

As shown, the three state charts have the same sequence of observable events applied and they generate the same outputs without any mismatch. This means that in the true model itself with V1SO, the effect of the unobservable event V1SO could not be determined or diagnosed, and it behaves as it has not happened. This aspect of undiagnosability made the nominal model unable to flag the discrepancy of the missing transition. This experiment concludes that the missing transitions with an undiagnosable unobservable failure event in the true model could not be detected and hypothesized.

3.6.1.7 Experiment U-7

In this experiment, missing more than one missing transition is studied. The plant will be comprised of two similar valves having stuck-open and stuck-closed fault modes. The automata of the components are shown in Figure 3.55. In this case, the nominal model will be built by deleting a transition in each of the two valves, i.e., missing two transitions. The two missing transitions are V1SC between C and SC in V1, and V2SC between C and SC in V2.

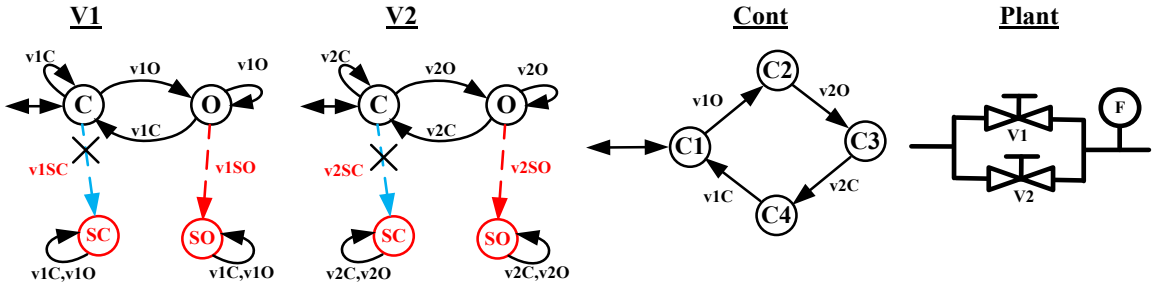


Figure 3.55: Experiment U-7 System

True model:

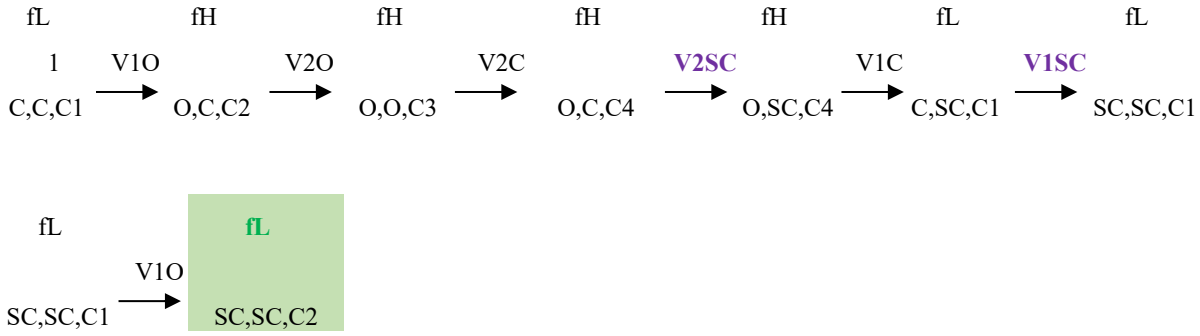


Figure 3.56: Experiment U-7 True Model

Nominal Model:

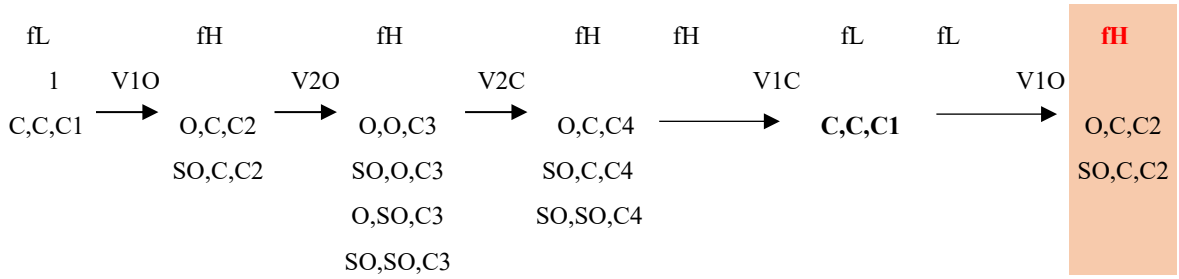


Figure 3.57: Experiment U-7 Nominal Model

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? **Wrong expected output after an observable event.**

   i. Expected State(ES)= {(O,C,C2), (SO,C,C2), (SO,O,C3),(O,SO,C3),}. By neglecting the controller component Cont, then ES become = {(O,C), (SO,C)}.

   a. Last Conforming State(LCS) in the Nominal model = (C,C,C1). By neglecting the controller component Cont, then LCS becomes = {(C,C)}

   c. Actual Label of Discrepancy (ALD) = fL.

2- **Step-2 Analysis:**

Suspected States table or the (SS-table) for the states in the model that can have the same output as Actual Label of Discrepancy **ALD** = **fL**.

SS-Table:

Table 3.28: Experiment U-7 SS-Table

V1	V2	Label
C	C	fL
SC	C	fL
C	SC	fL
SC	SC	fL

a. Reject the rows with all normal states (looking for faulty states).

Table 3.29: Experiment U-7 Filtered SS-Table

V1	V2	Label
SC	C	fL
C	SC	fL
SC	SC	fL

b. From the output map table, select the states that have some components conforming with Expected States (ES) = {(O,C), (SO,C)}.

For the first ES = (O,C), V2 is found to be C in the first record in the table (V1=SC, V2=C), thus (SC,C) is the corresponding *candidate expected state.*

For the second ES = (SO,C), V2 is found to be C one record in the table (V1=SC, V2=C), thus (SC,C) is the corresponding *candidate expected state.*

The summary Table 3.30 below summarizes the conclusion of sources and destinations of the missing transitions and is explain as follows:

Table 3.30: Experiment U-7 Source and Destination Summary Table

LCS	ES	Candidate Expected States	Suspected Component	Source	Destination
C,C	O,C	SC,C	V1	C	SC
	SO,C	SC,C	V1	C	SC

c. For the first ES = (O,C)

i. The candidate expected state (SC,**C**) when compared to ES = (O,**C**), we can find that V1 is wrongly expected, thus suspected components={V1}.

ii. Source of missing transition: in the Last Conforming State (LCS)=(C,C), where V1 is at C, then the source of missing transition in V1 is C.

iii. Transition Destination Candidate: from the found records (the candidate expected state) of the SS-table, V1 should be at SC, thus destination candidate in V1 is SC.

d. For the second ES = (SO,C)

i. The candidate expected state (SC,**C**) so, it is the same as the first ES = (O,C).

3- **Step-3: Hypothesize:**

a. Create a direct transition from the source to the candidate destination on the form:

b. Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

- For first expected state ES = (O,C)

H1= V1:C $\xrightarrow{V1SC}$ SC as SC is a state reachable by V1SC.

- For the second expected state ES = (SC,O).

H1= V1:C $\xrightarrow{V1SC}$ SC as SC is a state reachable by V1SC. But this hypothesis is **rejected** because it is repeated.

4- **Step-4: Update the model**

a. Re-combine all the components based on the new hypotheses in the suspected components.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

The same sequence of event will be applied then continued to check the future progression.

True model:

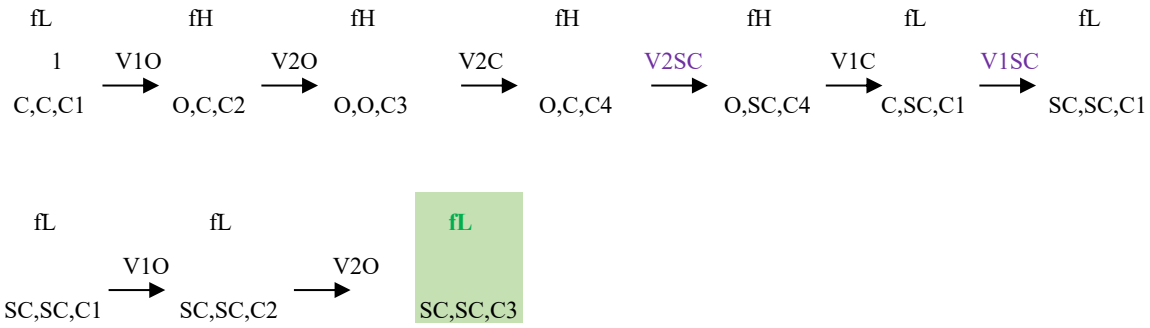


Figure 3.58: Experiment U-7 True Model

New Nominal Model:

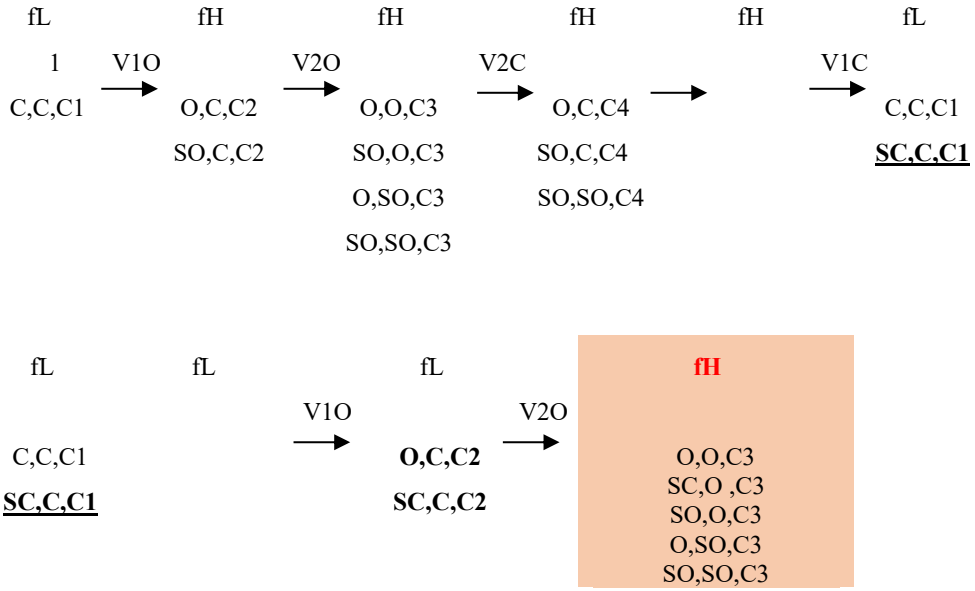


Figure 3.59: Experiment U-7 New Nominal Models

Notes:

- Practically, the nominal model does not see the unobservable events happening in the real system, but after correcting the previous discrepancy, V1SC is added to the model. If the previous discrepancy rehappens, it will be explained by V1SC, so the state flow diagram is updated accordingly including V1SC after the state (C,C,C1).
- Now, the nominal model is under the correction of hypothesis H1. A new discrepancy happens means either H1 is incompliant and needs to be rejected or this is already a new discrepancy. But, since the new nominal model is under a single hypothesis, the discrepancy is considered as a new discrepancy. This means for example, if the previous discrepancy leads to 3 hypotheses and the first of them has a discrepancy and the other two have not, the first will be considered incompliant while the other two surviving hypotheses will be considered compliant. If all hypotheses have a discrepancy, then this is considered a new discrepancy and needs to be corrected.
- Here in this case, there is only one hypothesis, and the new discrepancy needs to be corrected.

1- **Step-1 Information Collection**

a. Is there a discrepancy detected? Yes

b. Type of Discrepancy? **Wrong expected output after an observable event.**

j. Expected State(ES) = {(O,O,C3), (SC,O,C3), (SO,O,C3), (O,SO,C3),(SO,SO,C3)}.
For simplicity, we will study the first two expected states {(O,O,C3), (SC,O,C3)}.
By neglecting the controller component Cont, then ES becomes = {(O,O), (SC,O)}.

c. Last Conforming State(LCS) in the Nominal model: {(O,C,C2), (SC,C,C2)}. By neglecting the controller component Cont, then LCS becomes = {(O,C), (SC,C)}.

d. Actual Label of Discrepancy (ALD) = fL.

2- **Step-2 Analysis:**

Suspected States table or the (SS-table) for the states in the model that can have the same output as Actual Label of Discrepancy **ALD** = **fL**.

SS-Table:

Table 3.31: Experiment U-7 SS-Table (for New Nominal)

V1	V2	Label
C	C	fL
SC	C	fL
C	SC	fL
SC	SC	fL

- Reject the rows with all normal states (looking for states reachable by unobservable events which are fault events in this experiment).

Table 3.32: Experiment U-7 Filtered SS-Table (for New Nominal)

V1	V2	Label
SC	C	fL
C	SC	fL
SC	SC	fL

a. From the filtered SS-table, select the states that have some components conforming with Expected State (ES)= {(SC,O), (O,O)}.
For (O,O): No record was found, so, it will be assumed that both V1 and V2 are wrongly expected due to unobservable events that happened in both of them. Thus, the fully faulty states will be selected as candidate states. In this case (SC,SC) is the corresponding *candidate expected state.*

For (SC,O): V1 is found to be SC in two records in the table (V1=SC, V2=C), and (V1=SC, V2=SC) thus {(SC,C), (SC,SC)} are the corresponding *candidate expected states.*

The summary Table 3.33 below summarizes the conclusion of sources and destinations of the missing transitions and is explain as follows:

Table 3.33: Experiment U-7 Source and Destination Summary Table

LCS	**ES**	**Candidate Expected States**	**Suspected Component**	**Source**	**Destination**
(O,C)	(O,O)	(SC,SC)	V1	O	SC
			V2	C	SC
(SC,C)	(SC,O)	(SC,C)	---	---	---
		(SC,SC)	V2	C	SC

b. For the first ES = (O,O)

i. The candidate expected state (SC,S**C**) when compared to LCS = (O,**C**), we can find that V1 and V2 are wrongly expected, thus suspected components={V1,V2}.

- Source of missing transition: in the Last Conforming State (LCS)=(O,C), where V1 is at O, V2 is at C.

  Then the source of the first missing transition in V1 is O.

  The source of the second missing transition in V2 is C.

- Transition Destination Candidates:

  For the first candidate expected state (SC,SC),

  V1 should be at SC, thus the destination of first missing transition is SC.

  V2 should be at SC, thus the destination of first missing transition is SC.

c. For the second ES = (SC,O)

i. The first candidate expected state is neglected because it is the same as the LCS. This means that the missing transition will be as a self-loop. But the unobservable event such as faults should move the automaton from a normal state to a fault state.

ii. For the second candidate expected state (SC,S**C**), when compared to LCS = (SC,C), we can find that V2 is wrongly expected, thus suspected components={V2}.

iii. Source of missing transition: in the Last Conforming State (LCS)=(SC,C), where V2 is at C, then the source of missing transition in V2 is C.

iv. Transition Destination Candidate: from the found records (the candidate expected state) of the SS-table, V2 should be at SC, thus destination candidate in V2 is SC.

3- **Step-3: Hypothesize:**

a. Create a direct transition from the source to the candidate destination on the form:

b. Hypothesis = The suspected component: (Source → Discrepancy event → Destination Candidates).

- For first expected state ES = (O,O)

  H1= V1:O $\xrightarrow{\text{V1SC}}$ SC as SC is a state reachable by V1SC. This hypothesis is **rejected** because the cross fault between O and SC is not included in the setup of the true system.

  H2= V2:C $\xrightarrow{\text{V1SC}}$ SC as SC is a state reachable by V2SC.

- For the second expected state ES = (SC,O).

  H3= V2:C $\xrightarrow{\text{V2SC}}$ SC as SC is reachable by V2SC, but this hypothesis is **rejected** but this hypothesis is rejected because it is repeated in H2.

4- **Step-4: Update the model**

c. Re-combine all the components based on the new hypotheses in the suspected components.

d. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

e. The missing transition has been retrieved the same as the true model, so no need to verify the progression.

3.6.1.8 Experiment U-8

In this experiment, missing more than one missing transition is studied. The automata of the components are shown in Figure 3.60. In this case, the nominal model will be built by missing a transition in each of the two valves. The two missing transitions are V1SO between C and SO in V1, and V2SO between C and SO in V2.

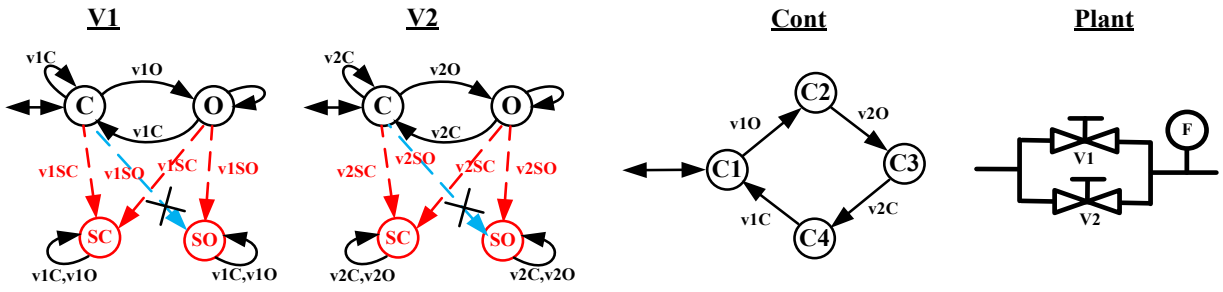
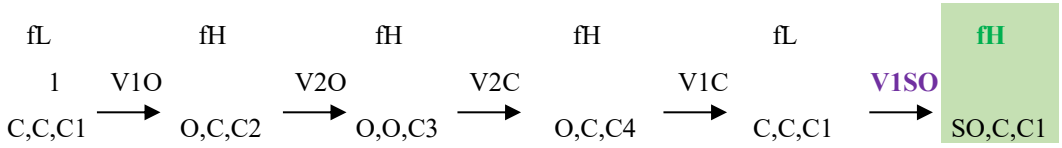


Figure 3.60: Experiment U-8 System

True model:



Nominal Model:

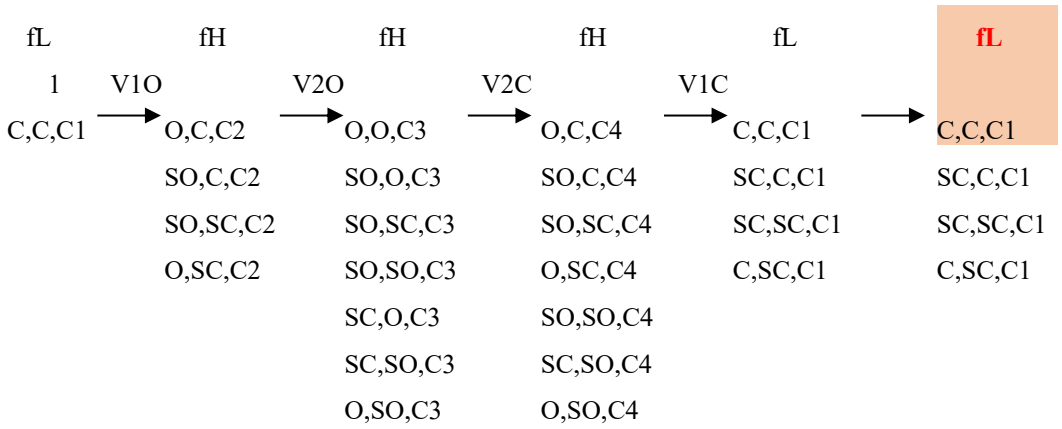


Figure 3.61: Experiment U-8 True and Nominal Models

1- **Step-1 Information Collection**

   a. Is there a discrepancy detected? Yes

   b. Type of Discrepancy? **Wrong unexpected output.**

   Note: Here the nominal model was matching the system until it reached the state (**C,C,C1**) with output=**fL** and suddenly the real system generated a new different output **fH** while the currently expected output is **fL**. This sudden change happened due to an unobservable event that could not be explained by the nominal model.

   Expected State (ES) after neglecting the controller component = {(C,C), (SC,C), (SC,SC,), (C,SC)}.

   a. Last Conforming State(LCS) in the Nominal model after neglecting the controller component = {(C,C), (SC,C), (SC,SC,), (C,SC)}.

   b. Actual Label of Discrepancy (ALD) = fH.

2- **Step-2 Analysis:**

Suspected States table or the (SS-table) for the states in the model that can have the same output as Actual Label of Discrepancy **ALD** = **fL**.

SS-Table:

Table 3.34: Experiment U-8 SS-Table

V1	V2	Output
O	O	fH
O	C	fH
O	SO	fH
O	SC	fH
SO	O	fH
SO	C	fH
SO	SO	fH
SO	SC	fH
C	O	fH
SC	O	fH
C	SO	fH
SC	SO	fH

a. Reject the rows with all normal states (looking for faulty states).

Table 3.35: Experiment U-8 Filtered SS-Table

V1	V2	Output
O	SO	fH
O	SC	fH
SO	O	fH
SO	C	fH
SO	SO	fH
SO	SC	fH
SC	O	fH
C	SO	fH
SC	SO	fH

Here in this experiment, and because the LCS and ES have four elements, the explanation will be very long. Therefore, using the same steps made in the previous experiments, we will generate the hypotheses based on the summary Table 3.36 below that summarizes the conclusion of sources and destinations of the missing transitions.

Table 3.36: Experiment U-8 Source and Destination Summary Table

LCS	ES	Candidate Expected States	Suspected Component	Source	Destination	Hypothesis Selection Reason
(C,C)	(C,C)	(C,SO)	V1	C	SO	Kept[1].
		(SO,C)	V2	C	SO	Kept[1].
(SC,C)	(SC,C)	(SC,O)	V2	C	O	Rejected[2]: Ends with normal states reachable by observable event.
		(SC,SO)	V2	C	SO	Kept[1].
		(SO,C)	V1	SC	SO	Rejected[3]: Fault mode is Permanent.
(SC,SC)	(SC,SC)	(O,SC)	V1	SC	O	Rejected[3]: Fault mode is Permanent.
		(SO,SC)	V1	SC	SO	Rejected[3]: Fault mode is Permanent.
		(SC,O)	V2	SC	O	Rejected[3]: Fault mode is Permanent.
		(SC,SO)	V2	SC	SO	Rejected[3]: Fault mode is Permanent.
(C,SC)	(C,SC)	(O,SC)	V1	C	O	Rejected[2]: Ends with normal states reachable by observable event.
		(SO,SC)	V1	C	SO	Kept[1]
		(C,SO)	V2	SC	SO	Rejected[3]: Fault mode is Permanent.

Notes:

[1]In the summary table, these hypotheses are sourcing from a normal state and ends with a destination state reachable by unobservable event. Thus, it is considered logically valid where we are treating missing transitions with unobservable events.

[2] These hypotheses are rejected because the destination is a normal state reachable by observable event.

[3] These hypotheses are rejected because they are sourcing from a fault mode state and the destination is different while it is assumed in this thesis that the fault mode is permanent.

3- <u>Step-3: Hypothesize:</u>

a. Create a <u>direct</u> transition from the source to the candidate destination on the form:

b. <u>Hypothesis</u> = The suspected component: (Source → Discrepancy event → Destination Candidates).

- For the expected state ES = (C,C)

  H1= V1:C $\xrightarrow{V1SO}$ SO as SO is a state reachable by V1SO.

  H2= V2:C $\xrightarrow{V1SO}$ SC as SO is a state reachable by V2SO.

- For the second expected state ES = (C,SC).

  H3= V1:C $\xrightarrow{V1SO}$ SO as SO is a state reachable by V1SO, but this hypothesis is **rejected** but this hypothesis is rejected because it is repeated in H1.

4- **Step-4: Update the Model**

a. Re-combine all the components based on the new hypotheses in the suspected components.

b. Check the future progression of each hypothesis according to the actual labels initiated by the real system and reject the incompliant.

True model:

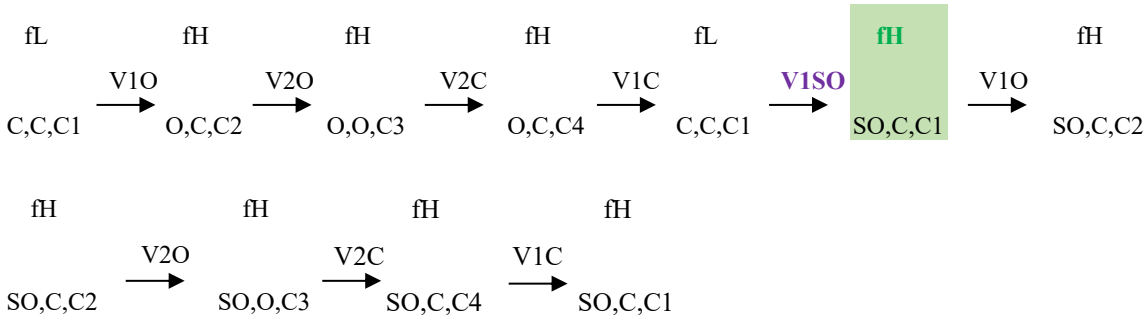


Figure 3.62: Experiment U-8 True Model

For simplicity, here the progression of the nominal models generated by H1 and H2 will not show all the possible states after each event and will only show the state reached by the event on the transition.

Nominal Model: Built by using H1

The model reaches C,C,C1 after V1O,V2O,V2C,V1C, then the output changes to fH. This model has an explanation for this change which is V1SO after V1C.

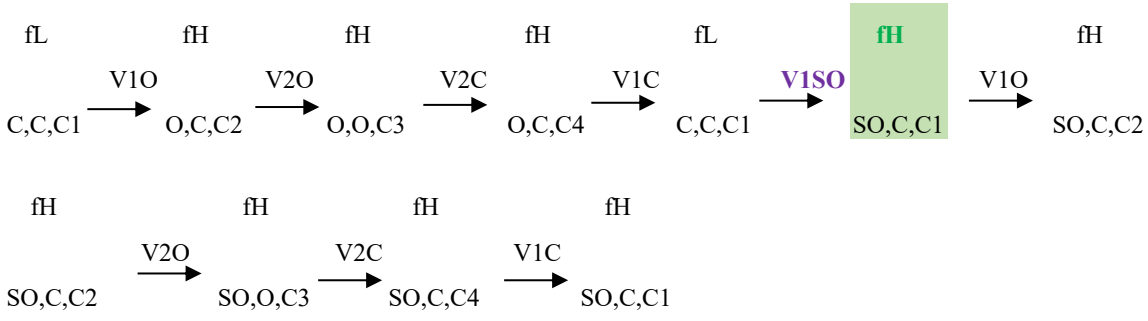


Figure 3.63: Experiment U-8 Nominal Model by H1

Nominal Model: Built by using H2

The model reaches C,C,C1 after V1O,V2O,V2C,V1C, then the output changes to fH. This model has an explanation for this change which is V2SO after V1C.

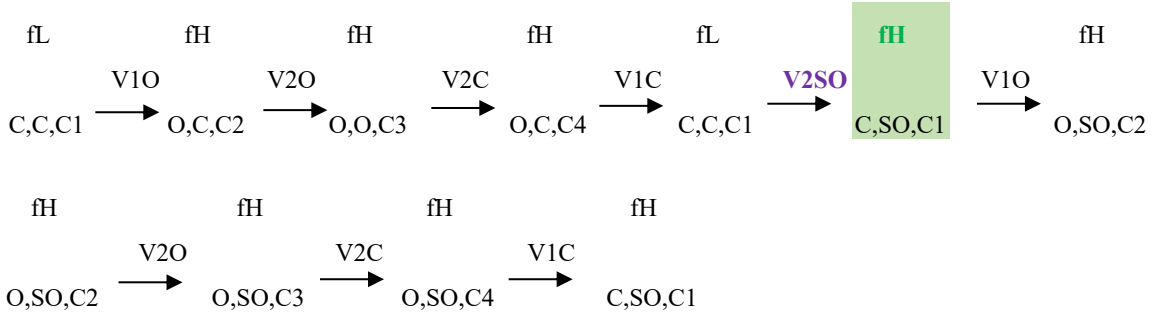


Figure 3.64: Experiment U-8 Nominal Model by H2

Discussion:

As shown above, the two hypotheses H1 and H2 are equal in terms of generating the same output response under the same sequence of observable events.

By repeating the future progression check with changing the sequence of event after the instant of discrepancy, it could be noted that the behavior of the two hypotheses is the same. This is since any of the valves is stuck open the output label will be fH always. Although the nominal model at the beginning of this experiment was built by removing two transitions, retrieving one of them is sufficient to explain the discrepancy. In other words, in H1, retrieving the missing V1SO between C and SO in V1 makes the missing V2SO between C and SO in V2 to be undiagnosable and consequently cannot be covered. And the same is correct for H2.

In such configuration, if we found two hypotheses having the same behavior under all possible sequence of events, we can consider them equal hypotheses where we can select any of them and reject the other or combine them based on the designer's decision.

This could be formulated as:

In this experiment, the two hypotheses are generated in two different components V1 and V2, thus it could be possible to merge these equal hypotheses so that each component will have its own missing transition corrected. In case of the two hypotheses are generated on the same component, for a demonstration example if we assumed there is two hypotheses as:

H1= V1:C $\xrightarrow{V1\sigma}$ O and H2= V1:C $\xrightarrow{V1\sigma}$ Po, then the two hypotheses shall not be merged, and they would be checked separately for future progression. This is since merging the two hypotheses will lead to a

nondeterministic model having two transitions with the same source and event but with different destinations. This case is out of scope of the assumptions of this research that tackles the missing transitions in deterministic models.

3.6.2 Derived Rules for the Learning Algorithm

The following rules are concluded from the previous experiments and will be used in constructing the algorithm that corrects the model's missing transitions with unobservable events. The rules are named as UnObRule-n, where the prefix UnOb stands for UnObservable events while n is the rule number.

UnObRule-1: The discrepancy of missing transition of unobservable event is detected by having unexpected output that could not be explained by the current nominal model.

- **UnObRule-1.a:** Type-1 is the unexpected output after an observable event is applied after the Last Conforming State.
- **UnObRule-1.a:** Type-2 is the unexpected output that suddenly happens without any observable event is applied after the Last Conforming State.

UnObRule-2: Expected state (ES) is the state in nominal model at which the output discrepancy occurs.

UnObRule-3: The suspected states table (SS-table) lists all the states having the same output as ALD. The fully normal states (reachable by observable events) are rejected.

UnObRule-4: The *candidate expected state*s are concluded from SS-table to have some components conforming with Expected State (ES).

- **UnObRule-4.a:** Missing transition with unobservable event should end with a state that is reachable by unobservable event in the component.
- **UnObRule-4.b:** Since within the context of this thesis, it is assumed that the fault mode is permanent, if the source state has a component in a faulty state (reachable by unobservable fault event), this component should not change its state.
- **UnObRule-4.C:** The suspected components are the wrongly expected components.

UnObRule-5: The Last Conforming State (LCS) gives the source of the missing transitions in suspected components.

UnObRule-6: The destination of the missing transition in the suspected component is the value of this component in the candidate expected states.

UnObRule-7: a hypothesis of a missing transition starts from a source to a destination with the applicable unobservable event that can reach the destination.

UnObRule-8: The number of hypothetical models equals the number of hypotheses of the missing transitions.

UnObRule-9: In terms of the future progression of the system, the compliant models are accepted, and the incompliant models are rejected.

UnObRule-10: Equal hypotheses are reduced to one hypothesis per transition if they pertain to the same component so that they could be tested separately. If the Equal hypotheses pertain to different components, they can be merged in a combined hypothesis.

Chapter 4

The Proposed Learning Algorithm

This chapter introduces the design of the diagnoser and the heuristic algorithm that is proposed as a solution for the previously discussed problem of incomplete models. As discussed in chapter 3, this study includes a number of experiments in order to explore the possible cases of the missed transitions. The rules follow the procedures used in the experiments of chapter 3. Since the algorithm should be triggered by a discrepancy, it is required to introduce the design of the diagnoser that monitors the system and detects the discrepancy. The steps of the learning algorithm are developed in such a way to first capture the current information of the system at the discrepancy. Then, the analysis of these information leads to the determination of the suspected components and the states at which the missing transitions are possibly missing. Finally, the hypotheses proposing the missing transitions are used to generate new models. The new model is verified based on its compliance with future behavior to the true system. The compliant hypotheses that make the nominal model generate the same response as the true system shall be accepted meanwhile the incompliant hypotheses shall be rejected.

4.1 Event-State Observer Design

4.1.1 Introduction

This section introduces the event-state based observer including its update laws and cases of detecting the discrepancy along with examples of how it works. While the observer will be used here for monitoring the progression of the model, the diagnosis will be carried out by the diagnoser block.

Two major approaches of observers or diagnosers are tackled in the literature. The event-based approach uses only the events with standard automata in the design of the diagnosers as per [18], [19]. The other approach of diagnoser is the state-based diagnoser that uses the output labels as inputs for the diagnoser to estimate the system state and the system condition [20]. Some other research combined the event-state based diagnoser in a timed version such as [28].

In this research, we represent the incomplete nominal model that needs to be corrected as a version of

Moore automaton having events on the transition arcs and the output labels on the states. Therefore, both of the two pieces of information which are the events and output labels will be incorporated in our observer block. This means that in this research, the observer will be an Event State-Based Observer. We assume that the observer block shall be initialized at the same time as the true system is initialized. As inputs, the observer receives the observable events generated in the plant and all output changes. Based on the received inputs either observable event, output change or both, the observer will compute the estimated states of the system. The state estimates are a set of states of the model where the observer informs that the system is estimated to be in one of those states.

The observer is an automaton designed based on the current model [20], then, if the model is incomplete and the observer at a certain point was unable to estimate the system states and evaluates to an empty set, this condition will be signaled as a discrepancy. The observer was explained in chapter 2 in the background overview however, Example 4.1 is introduced to recall and facilitate the idea, and the full discussion of the observer structure is given in the following section.

Example 4.1: As shown in Figure 4.1, assuming that the system is at state x_3 and an observable event σ_1 occurred, so the nominal model says that the system shall progress to state x_4. But since there are some transitions in the nominal model after x_3 having unobservable events α_1 and α_2 such that they might happen while they are unmeasurable. Thus, the observer shall estimate that when the system was at x_3 and σ_1 occurred, the system is estimated to be in any of the states $\{x_4 , x_{11} , x_{12}\}$. If σ_2 occurs, the system is estimated to be in $\{x_5 , x_{13}\}$.

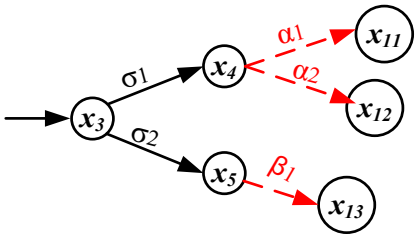


Figure 4.1: Example 4.1 Nominal Model

More formally, if we assumed that at certain instant k, the observer state estimate was $z_k = \{x_3\}$, then after σ_1 the new state estimate will be $z_{k+1} = \{x_4 , x_{11} , x_{12}\}$. Or at $z_k = \{x_3\}$, if σ_2 occurs the new state estimate $z_{k+1} = \{x_5 , x_{13}\}$ because after x_5 the unobservable event β_1 might happen. Figure 4.2 depicts the observer. Now, we can say that z_1 , z_2 and z_3 are the states of the observer automaton with each of them contains the state estimates of the nominal model based on the occurred event. It should be noted

that the observer in this example is an event-based observer.

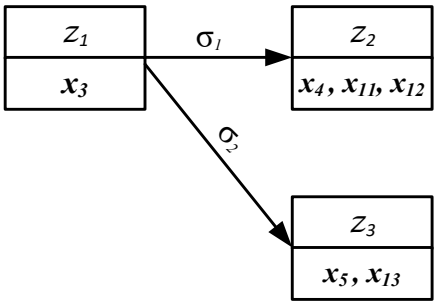


Figure 4.2: Example 4.1 Observer Automaton

4.1.2 Plant Model

The plant is modeled with best of knowledge to be the nominal model. Thus, the nominal model could be represented in terms of the following tuple.

$$G=(X,\Sigma,\eta,x_0,Y,\lambda,X_m),$$

Where X is the set of states, Σ: finite set of events, $\eta: X \times \Sigma \rightarrow X$ is the partial transition function, x_0 is the initial state, Y is the finite set of output labels, $\lambda: X \rightarrow Y$ output map function, X_m is the finite set of marked states.

4.1.3 Definitions

The ***unobservable reach*** of a state x denoted as ***UR(x)*** introduced in [1] is reintroduced here.

Definition 4.1 The unobservable reach of a state $x \in X$ is defined as follows:

$UR(x) = \{v \in X \mid (\exists\, t \in \Sigma_{uo}^{*})$ and $\eta\,(x,t) = v\}$.

□

Example 4.2 In Example 4.1 , $UR(x_4) = \{x_4, x_{11}, x_{12}\}$.

In addition, this definition could be extended to a set of states $z \subseteq X$ as:

$UR(z) = \cup_{x \in z}\, UR(x)$

Example 4.3 To obtain the unobservable reach for the set of states $\{x_4, x_5\}$, in Figure 4.1, let

$z = \{x_4, x_5\}$, and then $UR(z) = UR(x_4) \cup UR(x_5) = \{x_4,x_{11},x_{12}\} \cup \{x_5,x_{13}\} = \{x_4,x_5,x_{11},x_{12},x_{13}\}$

Definition 4.2 For any two states $x_1, x_2 \in X$, we say that x_2 is ***e-Reachable*** from x_1 if $x_1 \neq x_2$ and x_2 is reachable from x_1 by an unobservable event and is denoted as $x_1 \xrightarrow{\sigma \in \Sigma_{uo}} x_2$.

□

It would be required to extend the transition function η of the model's automaton such that it could be applied on a set of states rather than one state. Thus, for $z \subseteq X$, and $\sigma \in \Sigma$, $\eta(z,\sigma) = \{x \in X \mid (\exists\ x_e \in z)$ and $x=\eta(x_e ,\sigma)\}$, this means that it will result in the set of states reached by the event σ starting from the set of states z.

4.1.4 Observer Model

The observer automaton is represented by the following tuple:

$$O = (Z, \Sigma, Y, \eta, \lambda, z_0, \xi_\sigma, \xi_y, \xi_e)$$

where,

Z: set of states of the observer $z \subseteq 2^X - \{\phi\}$.

Σ : Set of events of the system.

Y : Set of output labels of the system.

$\eta: X \times \Sigma \rightarrow X$, the partial transition of the nominal model.

$\lambda : X \rightarrow Y$, the output map function.

z_0 : the initial state estimate of the observer.

Transition function: is also called the ***update law*** since it is the function used to update the observer from its current state to the next state.

$\xi_\sigma : Z \times \Sigma \rightarrow Z$, is the transition function (*update law*) of **case-1** of receiving an observable event without having an output label change. (Will be discussed in detail in section 4.1.5.1).

$\xi_y : Z \times Y \rightarrow Z$, is the transition function (*update law*) of **case-2** of having an output label change without receiving an observable event . (Will be discussed in detail in section 4.1.5.24.1.5.1).

$\xi_e : Z \times \Sigma \times Y \rightarrow Z$, the transition function (*update law*) of **case-3** of receiving an observable event with

an output change.(Will be discussed in detail in section 4.1.5.3).

As mentioned earlier, we assume that the observer and the true system will be initialized at the time. This implies that initial state $Z_0 = \{x \mid x \in UR(\{x_0\}) \text{ and } \lambda(x) = \lambda(x_0)\}$.

4.1.4.1 Observer State Notation

Figure 4.3 shows the notation of the observer state that is comprised of the three fields. The top field holds the observer state number. The middle field shows the estimated states of the nominal model. The bottom field holds the corresponding output label generated by the true system. The entering arrow represents the observation that leads to this state of observer (either the observable event or, the output label or, both of them).

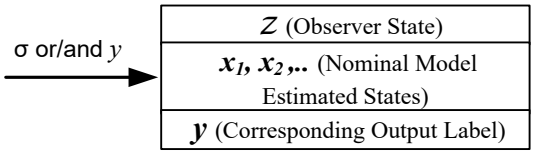


Figure 4.3: Observer State Notation

4.1.5 Cases of Observer Transition Function

The observer block is introduced as a Moore automaton that has a transition function which defines how the automaton states changes. Here, this automaton of the observer will have a transition function with three rules based on the case of the occurrence of the events and outputs. Briefly, the three cases of the transition function are as follows.

1- In the first case, a new observable event occurs in the system that causes **no change** in the output label.

2- In the second case, there is a change in the output label **without** an observable event.

3- In the third case, a new observable event occurs in the system that causes **a change** in the output label.

Figure 4.4 depicts the construction of the event-state observer in terms of the three cases of transition function.

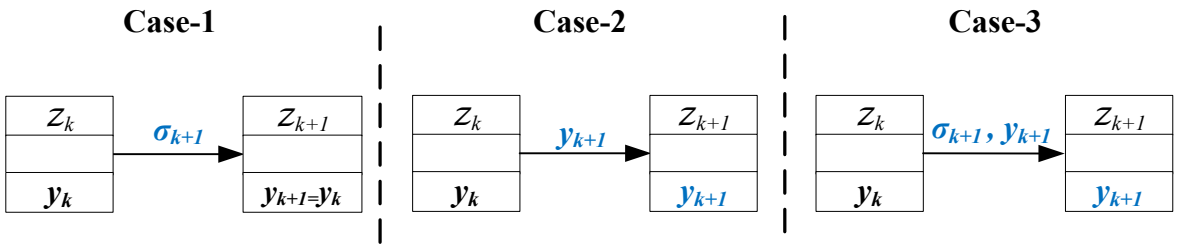


Figure 4.4: Observer Transition Function Cases

The transition function cases are formally described below, while some examples of how the observer operates in these cases are given in the next section.

4.1.5.1 Transition Function Case-1

In this case, the true system generates a new observable event σ_{k+1} but there is no change in the output label. The transition function (*update law*) of this case is defined as follows:

$z_{k+1} = \xi_\sigma (z_k, \sigma_{k+1})$ where,

$\xi_\sigma (z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1})) \text{ and } \lambda(x)=y_k\}$.

Figure 4.5 depicts the idea by an example. Here it is assumed that $z_k=\{x_{e1}, x_{e2}, x_{e3}\}$ is a certain state of the observer, and after the event σ_{k+1} occurs, the element states $\{x_{e1}, x_{e2}, x_{e3}\}$ evolved to $\{x_{i1}, x_{i2}, x_{i3}\}$, then the new observer state estimation z_{k+1} is the Unobservable Reach of $\{x_{i1},x_{i2},x_{i3}\}$ having an output label y_{k+1} where $y_{k+1} = y_k$.

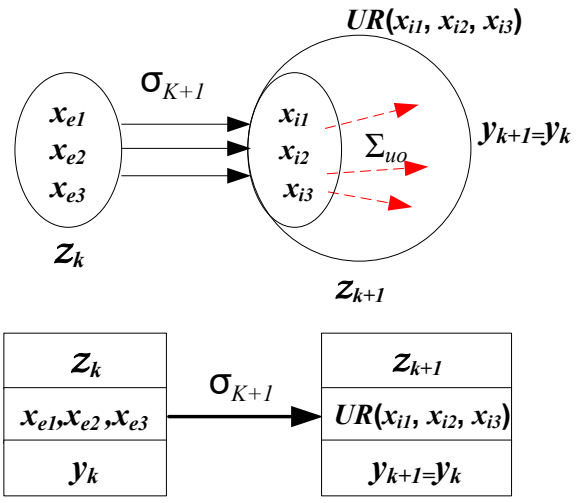


Figure 4.5: Observer Transition Function Case-1

This is an adapted version of the event-based observer transition function introduced in [1]. In other words, it could be said that the result of the event-transition function is to return all the states that could be reached from the set of states z_k by a sequence of events that starts with σ_{k+1} followed by unobservable events and all such states have the same output $y_{k+1} = y_k$.

4.1.5.2 Transition Function Case-2

In this case, the true system generates a new output without the occurrence of an observable event. The cause of this label change is an unobservable event. The transition function (*update law*) of this case is defined as follows:

$\mathcal{Z}_{k+1} = \xi_y(\mathcal{Z}_k, y_{k+1})$ where,

$\xi_y(\mathcal{Z}_k, y_{k+1}) = \{x \mid \lambda(x) = y_{k+1} \text{ and } x \in UR(\mathcal{Z}_k)\}$.

The transition function of this case is an adapted version of the state-based diagnoser transition function introduced in [20].

This means that, the new state estimation $\mathcal{Z}_{k+1}$ will be the set of states reachable from the previous $\mathcal{Z}_k$ by an unobservable event and having the new output label y_{k+1}.

Figure 4.6 depicts the idea by an example. If we assumed $\mathcal{Z}_k=\{x_{e1}, x_{e2}, x_{e3}\}$, then the new state estimate $\mathcal{Z}_{k+1}$ is the set of states reachable from $\mathcal{Z}_k$ by an unobservable event and having the new output label y_{k+1} generated by the true system.

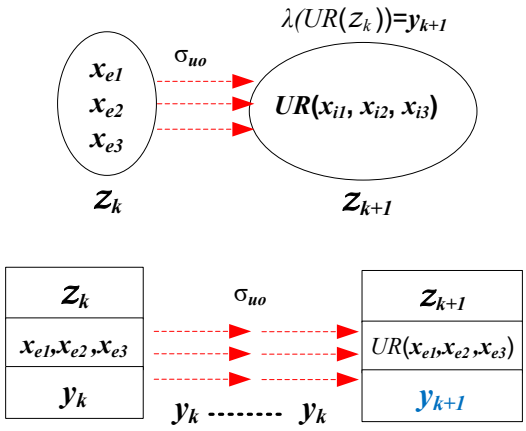


Figure 4.6: Observer Transition Function Case-2

4.1.5.3 Transition Function Case-3

In this case, a new observable event occurred in the system followed by a change of the output label. The transition function (*update law*) of this case is defined as follows:

$z_{k+1} = \xi_e (z_k, \sigma_{k+1}, y_{k+1})$ where,

$\xi_e (z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in \mathrm{UR}(\eta(z_k, \sigma_{k+1})) \text{ and } \lambda(x)=y_{k+1}\}$.

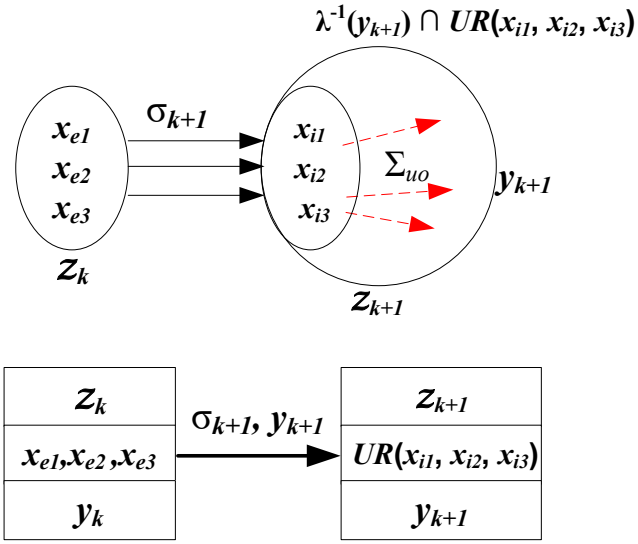


Figure 4.7: Observer Transition Function Case-3

Figure 4.7 depicts the idea that could be described as the first case but with having a new output label generated by the true system. When a new observable event σ_{k+1} occurs, the state estimate $z_k=\{x_{e1}, x_{e2}, x_{e3}\}$ can be updated to other states $\{x_{i1}, x_{i2}, x_{i3}\}$, then expanded to the unobservable reach with same output y_{k+1} to get the new estimate z_{k+1}.

It is worth to highlight that, in this research it is required to monitor the progression of the model in terms of two types of observations, the observable events and the output labels. Therefore, any discrepancy found that could not be explained by the current model shall be detected and forwarded to the learning algorithm.

4.1.6 Observer Block Operation Examples

The following examples are intended to describe how the observer works and also to show how

discrepancies will be detected.

4.1.6.1 Observer Block Example-1

This example introduces the observer block and ends with detecting a discrepancy of missing a transition with an observable event. The discrepancy in this example will be detected as an event happens in the true system but it is undefined in the nominal model at the discrepancy state. The plant is comprised of two cascaded valves. Figure 4.8 shows the automata of the plant. The first valve has fault modes of stuck-open and stuck-closed while the second valve is considered reliable with no faults. The ES stands for *Emergency Shut Down* command and is intended to be enabled on all the states of the two valves to let the valves go to their initial closed states. The observable transition with ES from state O to state C in V2 (denoted by the crossed line) is assumed to be missing in the model but it could happen in the true plant. This example will describe the progression of the model under the observation of the observer and shows how the missing transition is detected and declared as a discrepancy. The sequence of events is considered V1O, V2O, ES.

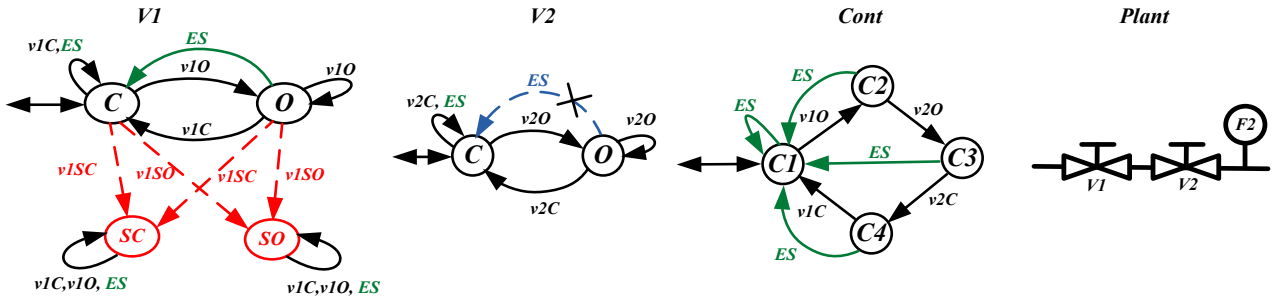


Figure 4.8: Observer Example-1 System

Figure 4.9 shows the automaton of the nominal model after performing the synchronous product of the components as G = sync (V1,V2,Cont). The sync operation is computed using ***Discrete Event Control Kit* (DECK)** [50] and plotted by MATLAB. The state name in the figure is depicted in terms of the state number and the components' values (e.g., 1:C,C,C1). The numbers assigned to states here are assigned by the sync operation in **DECK**. The transitions with unobservable events are indicated by the dashed lines while the solid lines represent observable events.

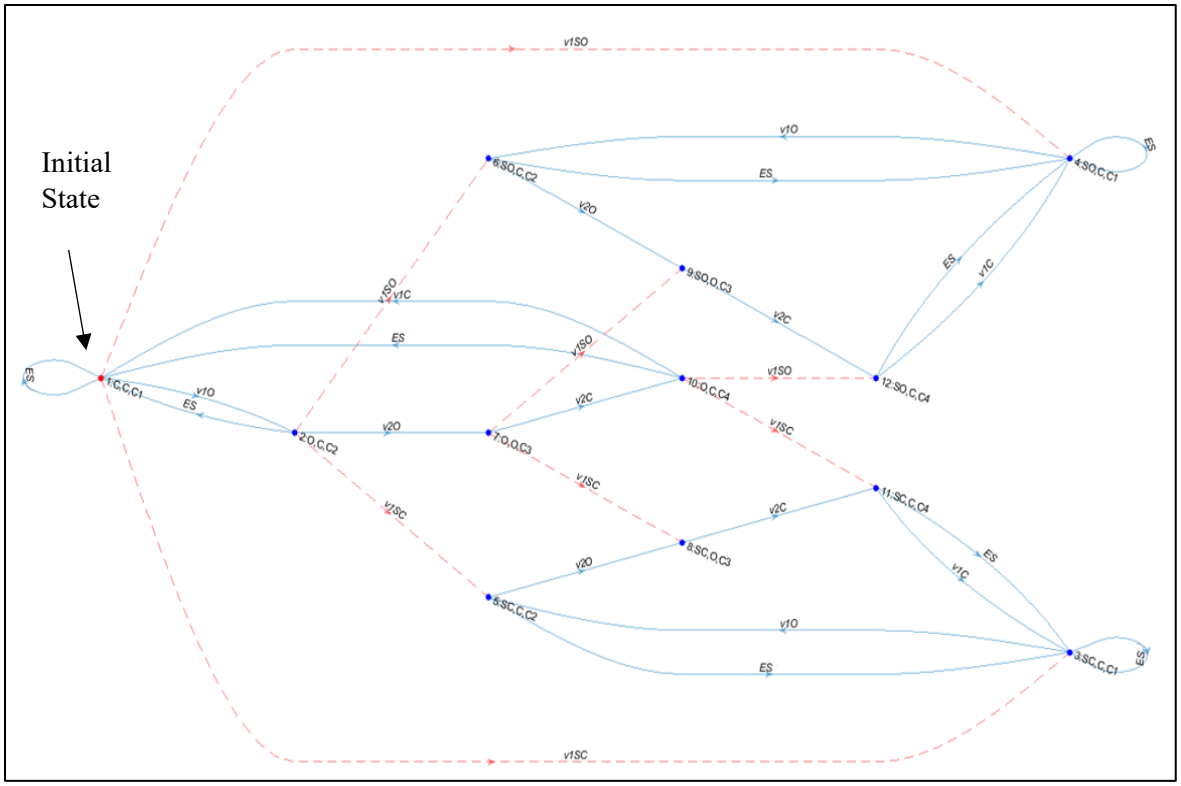


Figure 4.9: Observer Example-1 Nominal Model

Remarks about the examples:

- A new observation either a new observable event or a new output label will be denoted by the bold chevron bullet "➔" and will be followed by the corresponding transition computations.
- The diagnoser starts from its initial state z_0 that equals $UR(x_0)$. After a new observation (either a new observable event or a new output), the state estimate z_k will be updated by the observer's update law that corresponds to the case.

$z_0 = UR(x_0) = \{1,3,4\}$, is the initial state.

y_0 = fL , is the output label at the initial state.

➔ **V1O , y_1 = fL :** new event occurred at the real system with no label change (**Case-1**)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$

Then, $z_1 = \{x \mid x \in UR(\eta(z_0, \mathbf{V1O}))$ and $\lambda(x)=\mathbf{fL}\}$.

Since $z_0 = \{1,3,4\}$, $\eta(z_0, \mathbf{V1O}) = \{2,5,6\}$ as explained below.

$x \in z_0$	$\eta(x,\text{V1O})$	$UR(.)$	$UR(.)$ and $\lambda(x)$=fL
1	$\eta(1,\textbf{V1O})=2$	{2,5,6}	{2,5,6}
3	$\eta(3,\textbf{V1O})=5$	{5}	{5}
4	$\eta(4,\textbf{V1O})=6$	{6}	{6}

Thus, z_1= {2,5,6}.

➔ **V2O , y_2= fH:** new event occurred with label change (**Case-3**)

Since $z_{k+1} = \xi_e(z_k, \sigma_{k+1}, y_{k+1})$, $\xi_e(z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1})) \text{ and } \lambda(x)=y_{k+1}\}$.

Then, z_2= $\{x \mid x \in UR(\eta(z_1, \textbf{V2O})) \text{ and } \lambda(x)=\textbf{fH}\}$

Since z_1= {2,5,6}, $\eta(z_1, \textbf{V2O})$ = {7,8,9} as explained below.

$x \in z_1$	$\eta(x,\text{V2O})$	$UR(.)$	$UR(.)$ and $\lambda(x)$=fH
2	$\eta(2,\textbf{V2O})=7$	{7,8,9}	{7,9}
5	$\eta(5,\textbf{V2O})=8$	{8}	---
6	$\eta(6,\textbf{V2O})=9$	{9}	{9}

But $\lambda(8) \neq$ **fH**, thus, z_2 = {7,9}.

➔ ES, y_3= **fL:** new event occurred with label change (**Case-3**)

z_3= $\{x \mid x \in UR(\eta(z_2, \textbf{ES})) \text{ and } \lambda(x)=\textbf{fL}\}$

Since z_2= {7,9}, $\eta(z_2, \textbf{ES}) = \phi$ as explained below.

$x \in z_2$	$\eta(x, \text{ES})$	$UR(.)$	$UR(.)$ and $\lambda(x)$=fL
7	$\eta(7,\textbf{ES})$= NOT DEFINED	---	---
9	$\eta(9,\textbf{ES})$= NOT DEFINED	---	---

ES is NOT DEFINED at z_2 = {7,9}, **DISCREPANCY DETECTED**

The progression of the observer during the previous steps is depicted in Figure 4.10.

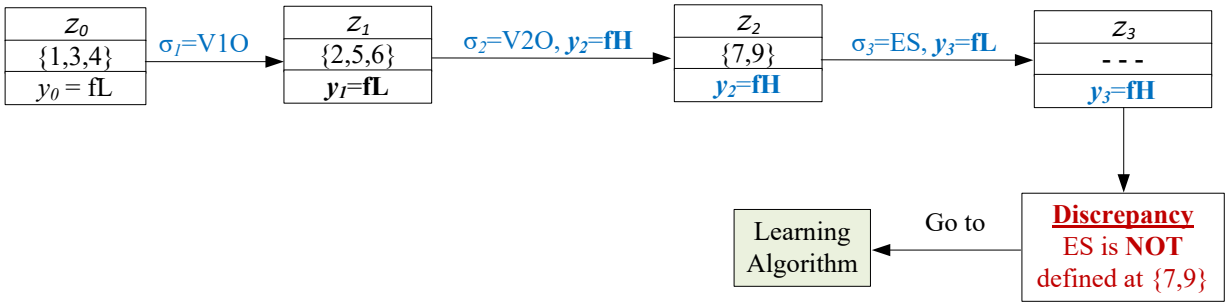


Figure 4.10: Observer Example-1 Progression

4.1.6.2 Observer Block Example-2

This example discusses the observer block in a case ending with detecting the discrepancy of missing a transition with an unobservable event. The discrepancy in this example will be detected as an unexpected output label generated and measured while the current nominal model is unable to explain it. This is due to an unobservable event happening in the true system, and its transition being absent in the current nominal model.

The plant is comprised of two valves as shown in Figure 4.11. The first valve has two modes of fault (stuck-open and stuck-closed) while the second one is considered without fault modes. The nominal model is built with missing the transition of the unobservable event V1SO between the state C and SO in V1.

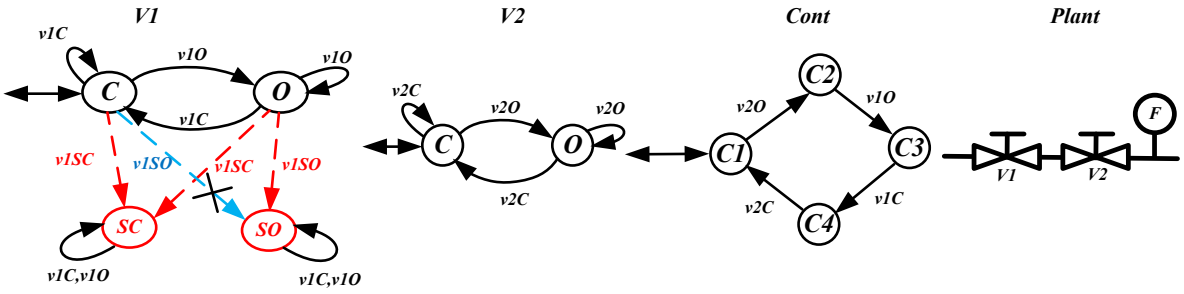


Figure 4.11: Observer Example-2 System

Figure 4.12 shows the automaton of the nominal model after performing the synchronous product of the components as G = sync (V1,V2,Cont).

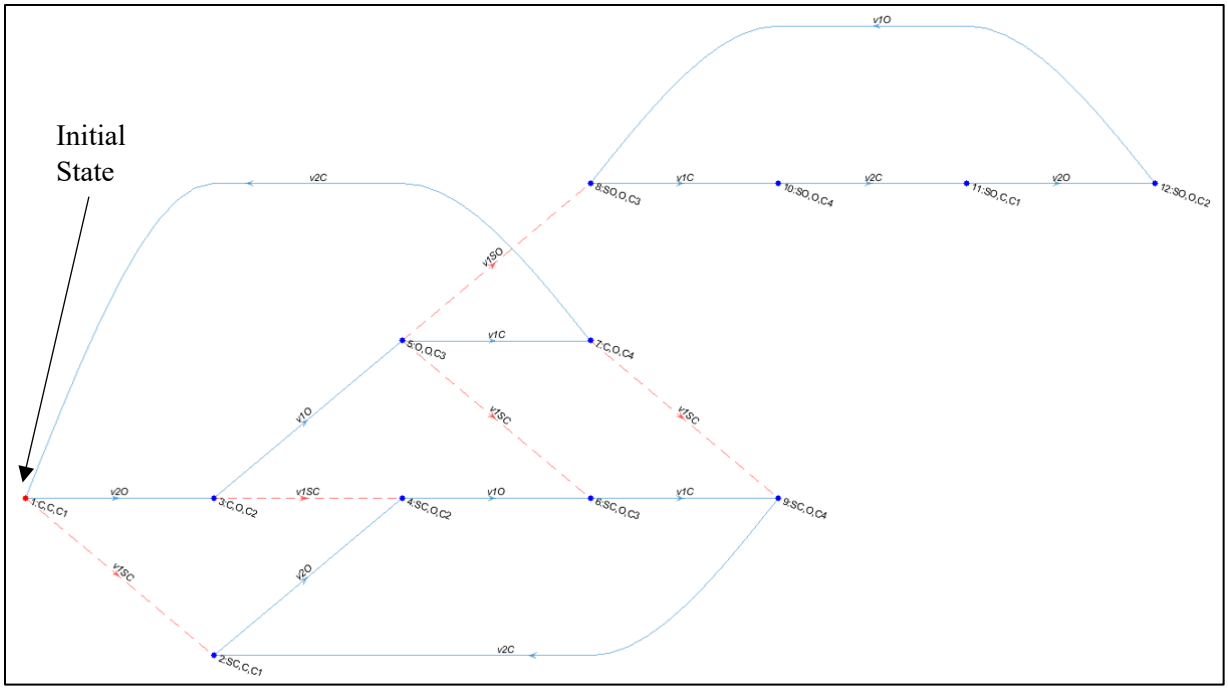


Figure 4.12: Observer Example-2 Nominal Model

$z_0 = UR(x_0) = UR(1) = \{1,2\}$, is the initial state.

$y_0 = \text{fL}$

➔ **V2O , y_1 = fL :** new event occurred at the real system with no label change (**Case-1**)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$.

Then, $z_1 = \{x \mid x \in UR(\eta(z_0\ \mathbf{V2O}))$ and $\lambda(x)=\mathbf{fL}\}$.

Since $z_0 = \{1,2\}$, $\eta(z_0, \mathbf{V2O}) = \{3,4\}$ as explained below.

$x \in z_0$	$\eta(x,\mathbf{V2O})$	$UR(.)$	$UR(.)$ and $\lambda(x)$=fL
1	$\eta(1,\mathbf{V2O})=3$	{3,4}	{3,4}
2	$\eta(2,\mathbf{V2O})=4$	{4}	{4}

Thus, $z_1 = \{3,4\}$.

➔ **V1O , y_2 = fH:** new event occurred with label change (Case-3)

Since $z_{k+1} = \xi_e(z_k, \sigma_{k+1}, y_{k+1})$, $\xi_e(z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_{k+1}\}$.

Then, $z_2 = \{x \mid x \in UR(\eta(z_1, \mathbf{V1O}))$ and $\lambda(x)=\mathbf{fH}\}$.

Since $z_1 = \{3,4\}$, $\eta(z_1, \mathbf{V1O}) = \{5,6\}$ as explained below.

$x \in z_1$	$\eta(x,\mathbf{V1O})$	$UR(.)$	$UR(.)$ and $\lambda(x)$=fH
3	$\eta(3,\mathbf{V1O})=5$	{5,6,8}	{5,8}
4	$\eta(4,\mathbf{V1O})=6$	{6}	---

But $\lambda(6) \neq$ **fH,** thus, $z_2 = \{5,8\}$.

➔ **V1C ,** y_3 **= fL:** new event occurred with label change (Case-3)

Then, $z_3 = \{x \mid x \in UR(\eta(z_2, \mathbf{V1C}))$ and $\lambda(x)=\mathbf{fL}\}$.

Since $z_2 = \{5,8\}$, $\eta(z_2, \mathbf{V1C}) = \{7,10\}$ as explained below.

$x \in z_2$	$\eta(x,\mathbf{V1C})$	$UR(.)$	$UR(.)$ and $\lambda(x)$=fL
5	$\eta(5,\mathbf{V1C})=7$	{7,9}	{7,9}
8	$\eta(8,\mathbf{V1C})=10$	{10}	---

But $\lambda(10) \neq$ **fL**, thus, $z_3 = \{7,9\}$.

Here, we can study the two scenarios related to unobservable events. The first scenario is the unexpected output label change and the second is the unexpected label change after an observable event.

Unexpected Output Label Change:

At the current situation, let us assume that the unobservable event V1SO occurs in the plant where the transition of V1SO from C to SO in V1 is missing in the nominal model. But it happens in the plant where V1 fails to stuck-open. This is an unobservable event and, it caused an output label change to **fH**.

➔ y_4 **= fH:** new label without an observable event (Case-2)

Since, $z_{k+1} = \xi_y (z_k, y_{k+1})$, where $z_{k+1} = \{x \mid \lambda(x) = y_{k+1}$ and $x \in UR(z_k)\}$

Then, $z_4 = \{x \mid \lambda(x) = \mathbf{fH}$ and $x \in UR(z_3)\}$, i.e., $z_4 = UR(z_3) \cap \lambda^{-1}(\mathbf{fH})$

From $z_3 = \{7,9\}$

$x \in z_3$	$UR(.)$	$UR(.)$ and $\lambda(x)$ = fH
7	{7,9}	---
9	---	---

This means that, for this label change, if we started from the states of $z_3 = \{7,9\}$ we cannot reach any state having label =fH by any unobservable events. Thus, the new label cannot be explained by the current model, and a **DISCREPANCY IS DETECTED**. Here a stuck-open failure from the closed state of V1 was not expected. The progression of the observer during the previous steps is depicted in Figure 4.13.

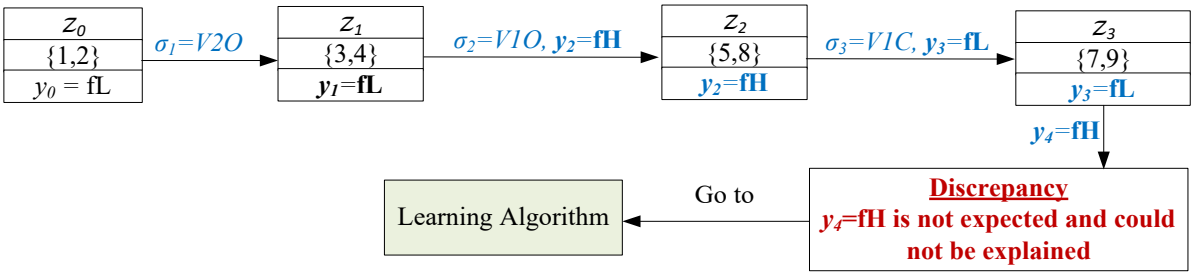


Figure 4.13: Observer Example-2 Progression

Wrong Expected Output After an Observable Event.

If we assumed that the system continued after $z_3 = \{7,9\}$ as follows:

➔ **V2C , y_4 = fL :** new event occurred at the real system with no label change (**Case-1**)

Then, $z_4 = \{x \mid x \in UR\ (\eta(z_3, \mathbf{V2C}))$ and $\lambda(x)$=**fL**$\}$.

Since, $z_3 = \{7,9\}$, $\eta(z_3, \mathbf{V2C}) = \{1,2\}$ as described below.

$x \in z_3$	$\eta(x,$**V2C**$)$	*UR*(.)	*UR*(.) and $\lambda(x)$=fL
7	η(7,**V2C**)=1	{1,2}	{1,2}
9	η(9,**V2C**)=2	{2}	{2}

Thus, $z_4 = \{1,2\}$.

➔ **V1SO unobservable event** occurs in the true system with its transition missing in the nominal model but since V2 is still closed, it will not cause any label change.

➔ **V2O**, y_5 **= fH**: new event occurred with label change (**Case-3**).
Note that, this label **fH** is due to V1 has failed to the stuck-open state by V1SO but the current nominal model expects the output label to remain **fL** since V1 is not yet commanded to open .

Then, $z_5 = \{x \mid x \in UR(\eta(z_4, \mathbf{V2O}))$ and $\lambda(x)$=**fH**$\}$.

For $z_4 = \{1,2\}$, we find $\eta(z_4,$**V2O**$)$ as described below.

$x \in z_4$	$\eta(x,$**V2O**$)$	*UR*(.)	*UR* and $\lambda(x)$=fH
1	η(1,**V2O**)=3	{3,4}	---
2	η(2,**V2O**)=4	{4}	---

Although the new event is V2O is defined at $z_4 = \{1,2\}$ but it leads to states {3,4} with output label **fL**, i.e., the nominal model expects to see an output label **fL**. This is while the plant actually generates a label **fH** that cannot be explained or expected by the current model. Thus, a **DESCRIPANCY IS DETECTED**. The progression of the observer during the previous steps is depicted in Figure 4.14.

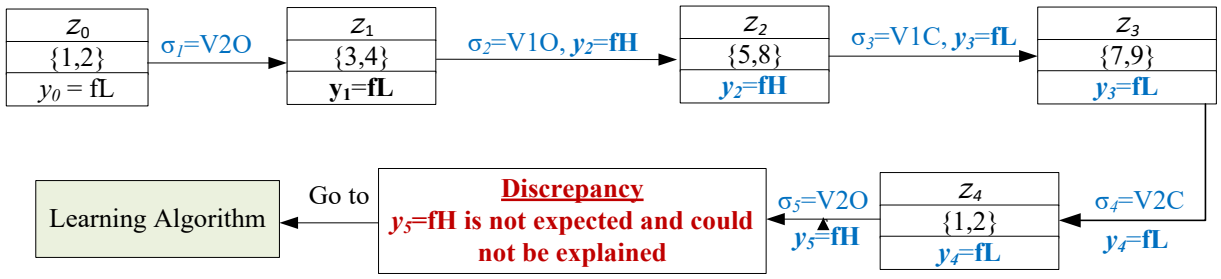


Figure 4.14: Observer Example-2 Progression

4.1.6.3 Observer Block Example-3

This example presents the operation of the observer block and discusses the case of unobservable events happening in the real system if they are already modeled in the nominal model.

The same configuration of the system used in Example-2 above will be used here with V1SO exists between C and SO.

$z_0 = \mathrm{UR}(x_0) = \{1,2\}$, is the initial state.

y_0 = fL, is the output label at the initial state.

➔ **V2O , y_1 = fL :** new event occurred at the real system with no label change (**Case-1**)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$.

Then, $z_1 = \{x \mid x \in UR(\eta(z_0$ **V2O**$))$ and $\lambda(x)=$**fL**$\}$.

Since $z_0 = \{1,2\}$, $\eta(z_0,$ **V2O**$) = \{3,4\}$ as explained below.

$x \in z_0$	$\eta(x,$**V2O**$)$	*UR*(.)	*UR*(.) and $\lambda(x)$=fL
1	$\eta(1,$**V2O**$)=3$	{3,4}	{3,4}
2	$\eta(2,$**V2O**$)=4$	{4}	{4}

Thus, $z_1 = \{3,4\}$.

➔ **V1O, y_2 = fH:** new event occurred with label change (Case-3)

Since $z_{k+1} = \xi_e(z_k, \sigma_{k+1}, y_{k+1})$, $\xi_e(z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_{k+1}\}$.

Then, $z_2 = \{x \mid x \in UR(\eta(z_1,$ **V1O**$))$ and $\lambda(x)=$**fH**$\}$.

Since $z_1 = \{3,4\}$, $\eta(z_1,$ **V1O**$) = \{5,6\}$ as explained below.

$x \in z_1$	$\eta(x \in z_1,$**V1O**$)$	*UR*(.)	*UR*(.) and $\lambda(x)$=fH
3	$\eta(3,$**V1O**$)=5$	{5,6,8}	{5,8}
4	$\eta(4,$**V1O**$)=6$	{6}	---

But $\lambda(6) \neq$ **fH,** thus, $z_2 = \{5,8\}$.

➔ **V1SO** occurs in the true system and since it is unobservable and under the current configuration, it will not cause any label change.

➔ **V1C, y_3 = fH:** new event occurred at the real system with no label change (Case-1)

Then, $z_3 = \{x \mid x \in UR\ (\eta(z_2, \mathbf{V1C}))$ and $\lambda(x)=\mathbf{fH}\}$.

Since $z_2 = \{5,8\}$, $\eta(z_2, \mathbf{V1C}))$ = {7,10} as explained below.

$x \in z_2$	$\eta(x,\mathbf{V1C})$	$UR(.)$	$UR(.)$ and $\lambda(x)$=fH
5	$\eta(5,\mathbf{V1C})=7$	{7,9}	---
8	$\eta(8,\mathbf{V1C})=10$	{10}	{10}

But $\lambda(7) \neq \mathbf{fH}$ and ,$\lambda(9) \neq \mathbf{fH}$, thus, thus, $z_3 = \{10\}$.

➔ **V2C , y_4 = fL:** new event occurred with label change (Case-3)

$z_4 = \{x \mid x \in UR(\eta(z_3, \mathbf{V2C})$ and $\lambda(x)=\mathbf{fL}\}$

Since $z_3 = \{10\}$, $\eta(z_3, \mathbf{V2C})$ = {11} as described below.

$x \in z_3$	$\eta(x,\mathbf{V2C})$	$UR(.)$	$UR \mid \lambda(x)$=fL
10	$\eta(10,\mathbf{V2C})=11$	{11}	{11}

Thus, $z_4 = \{11\}$.

The progression of the observer during the previous steps is depicted in Figure 4.15 in terms of the observable events.

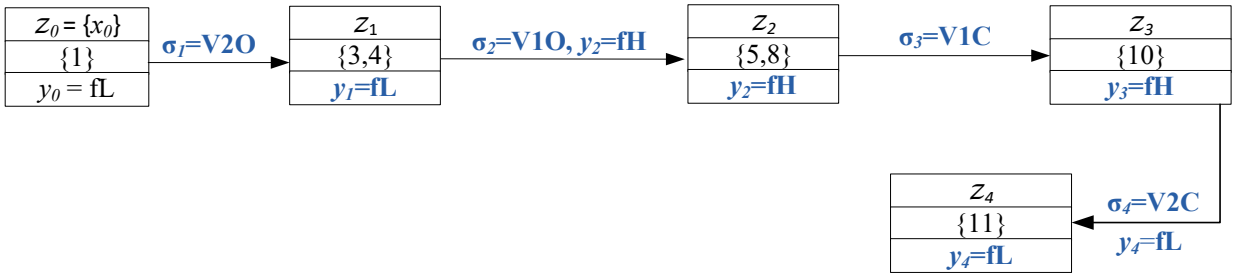


Figure 4.15: Observer Example-3 Progression

As shown above, the unobservable event V1SO could be explained by the nominal model and estimated by the observer based on the output label change.

4.2 The Learning Diagnoser Flowchart

Figure 4.16 shows the flow chart of the learning diagnoser. It is comprised mainly of two parts. The first part is the state-event observer that checks for new observations, updates state estimates and detects the discrepancy. If there is no discrepancy, the condition map function maps the estimated states to thier corresponding conditions (e.g., Normal, Fault). The second part is the learning algorithm that is triggered

when a discrepancy is detected. then, it works on generating hypotheses for the missing transitions.

The discrepancy is detected as soon as any of the three cases of update laws for the observer returns an empty state estimate. This means that the current nominal model cannot explain the observations from the true system, either the events or the actual output label. This inability to explain is due to the incompleteness of the model in terms of missing transitions in some components. Once the discrepancy is detected, the symptom of discrepancy shall identify its type and consequently, the type of the missing transition either a transition with observable event or unobservable event.

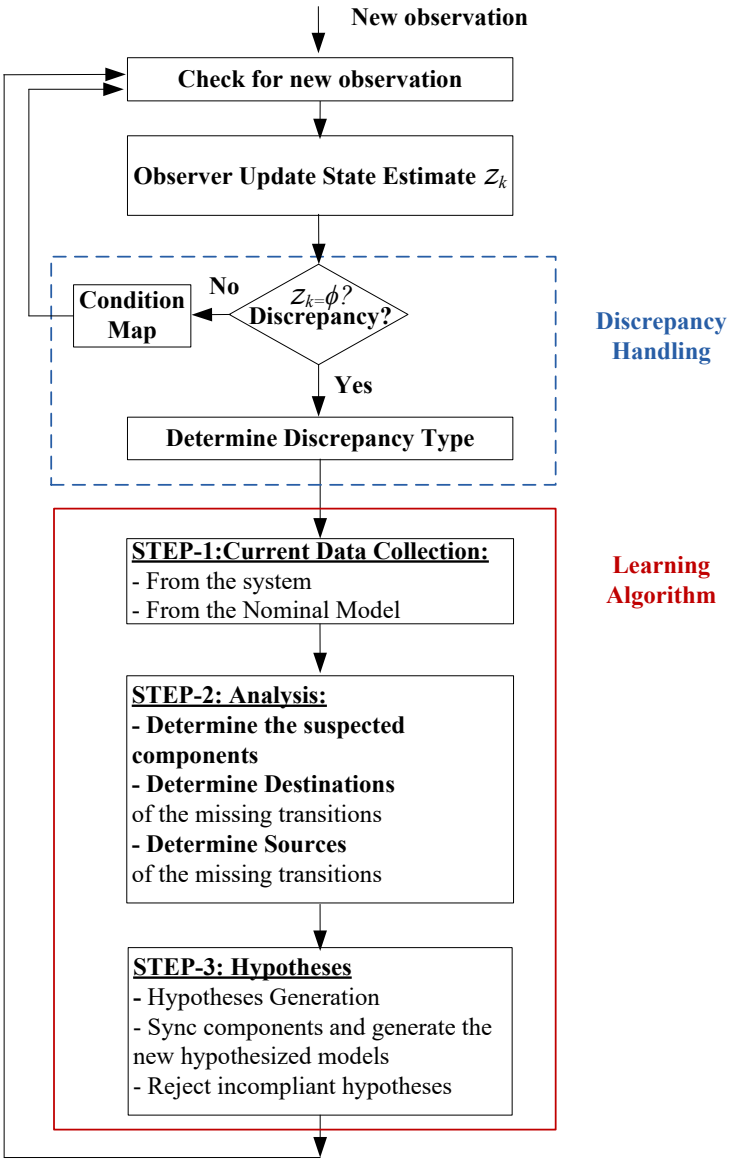


Figure 4.16: Learning Diagnoser Flowchart

4.3 Discrepancy Symptoms

In this study we have two types of missing transitions: (1) is missing transitions with observable events and (2) is, missing transitions with unobservable events. Each type of these missing transition generates discrepancy with its own symptoms.

4.3.1 Missing Transitions with Observable Events:

The discrepancy of this type of missing transitions in the components has one symptom that is denoted as "*Unexpected Observable Event*".

4.3.1.1 Unexpected Observable Event

The name comes from the situation that, when the true system is progressing and the observer is estimating, then a new observable event that is not defined at the last state estimate occurs. That is why it is not expected to be seen here at last state estimate. Hence, the observer update law evaluates an empty set ϕ and indicates a discrepancy. This unexpected observable event could be a private event that belongs to the alphabet of a single component, or it is a common event that belongs to the alphabets of more than one component.

4.3.2 Missing Transitions with Unobservable Events

The discrepancy of this type of missing transitions in the components has two symptoms that will be denoted as: (1) *Unexpected output label with observable event*, and (2) *Unexpected output label*.

4.3.2.1 Unexpected Output Label with Observable Event

The discrepancy symptom appears when an observable event occurs in the true system and generates an output label either the same as the previous one or a new one. This observable event is defined at the current state estimate of the observer, but it leads the nominal model to a different output label other than the actual label generated by the true system. This Unexpected output label generated in the true system happens due to a transition with unobservable event in the real system but was missing in the nominal model, and it is required to be added.

It should be noted that, the occurred observable event is already defined at the last conforming state estimate, so it could not be treated as a missing observable event. This is based on the theoretical definition of the synchronous product given in chapter.2, the event (common or private) is defined in the resulting

model automaton (at the current state) if it is defined in all automata - having the event in their alphabet - at their corresponding current states. Moreover, adding a new transition that sources from the last conforming state estimate holding this event (which has already this event going out of it), will make the automaton nondeterministic which is not the subject of this thesis.

4.3.2.2 Unexpected Output-Label

The name of this discrepancy symptom describes the situation in which the true system was at a certain state that matches the estimations of the observer and **without** any new observable event occurred in the true system, the true system suddenly generates an output label that cannot be explained by the current nominal model. Such sudden change is unexpected to occur by any observable event or a command in the true system.

4.4 Discrepancy Type Determination by the Observer

A discrepancy arises if after a new observation, the observer generates an empty state estimate. Depending on the last update law that was used, there are three cases of discrepancy. Therefore, this section explains how each update law is able to detect discrepancy symptoms.

4.4.1 While in Update Law of Case-1

In case-1, a new event occurred, and the output label does not change, The update law is:

$z_{k+1} = \xi_\sigma (z_k, \sigma_{k+1})$ where,

$\xi_\sigma (z_k, \sigma_{k+1}) = \{x \mid x \in UR (\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$.

Thus, the discrepancy handling block in the algorithm flow chart could be explained as follows:

- IF the term $\xi_\sigma (z_k, \sigma_{k+1})$ evaluates to an empty set ϕ (due to σ_{k+1} is not defined at the set of state estimate z_k because its transition is missing in some components),
  Then the type of discrepancy is ***unexpected observable event***. This depicted in Figure 4.17.

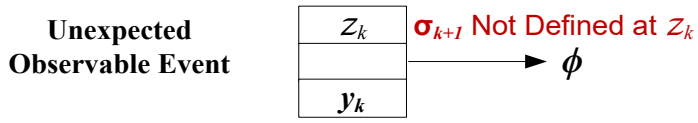


Figure 4.17: Unexpected observable Event from Case-1

Figure 4.18 explains this logic as a flowchart for Update law of **case-1**.

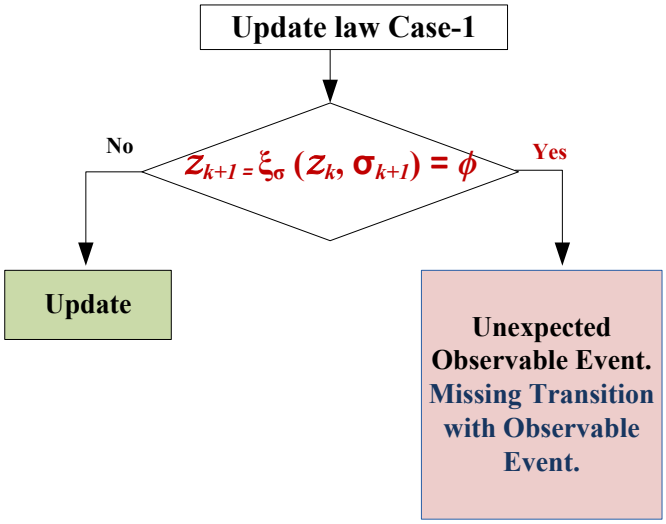


Figure 4.18: Discrepancy Detection by Case-1 Update Law

4.4.2 While in Update Law of Case-2

In case-2, the output label changes without any observable event, generated. The update law is:

$z_{k+1} = \xi_y (z_k, y_{k+1})$ where,

$\xi_y (z_k, y_{k+1}) = \{x \mid \lambda(x) = y_{k+1} \text{ and } x \in UR(z_k)\}$.

Thus, the discrepancy handling block in the algorithm flow chart could be explained as follows:

- IF the true system generates a new output label y' where $y' \neq y$ (the current output label) without an observable event and the new output label y' cannot be explained by the observer at the unobservable reach of the current state estimate, i.e., $y' \notin \lambda(UR(z_k))$,
  Then the type of discrepancy is ***unexpected output label***. This depicted in Figure 4.19.

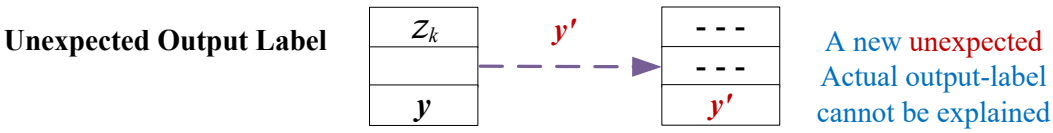


Figure 4.19: Unexpected Output Label

Figure 4.20 explains this logic as a flowchart for Update law of **case-2.**

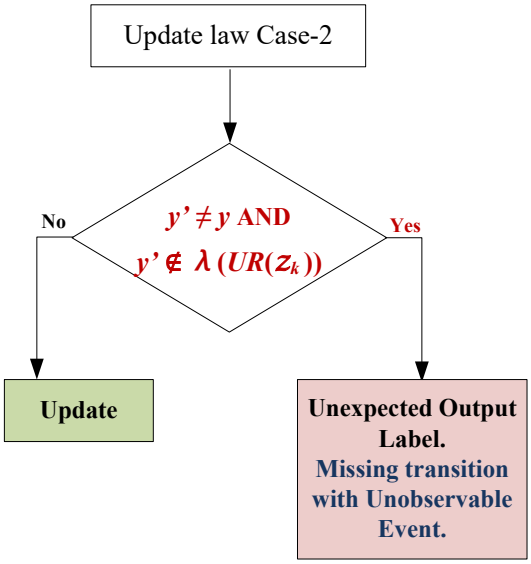


Figure 4.20: Discrepancy Detection by Case-2 Update Law

4.4.3 While in Update Law of Case-3

In case-3, a new observable event is generated along with output label change.

The update law is:

$z_{k+1} = \xi_e (z_k, \sigma_{k+1}, y_{k+1})$ where,

$\xi_e (z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x) = y_{k+1}\}$.

It should be noted that, this case could be regarded as an extension to case-1 (that has no output label change), but ***with*** output label change and it will be executed in two steps. The first step is to evaluate $UR(\eta(z_k, \sigma_{k+1}))$ if σ_{k+1} is defined at z_k, and the second step is to return the set of states, where $\lambda(x) = y_{k+1}$.

Hence, if the generated event in the true system is not defined at the last state, such that $\eta(z_k, \sigma_{k+1}) = \phi$, then the type of discrepancy is ***unexpected observable event***, (as detected by case-1). This is regardless of the value of the output label, because anyways the learning algorithm shall work on hypothesizing a transition with the missing observable event that explains the actual output label (either changed or not). However, if the generated observable event by the true system σ_{k+1} was defined at the last state z_k such that such that $\eta(z_k, \sigma_{k+1}) \neq \phi$, but the new actual output label y_{k+1} cannot be explained by the model, it

means that an unobservable event could have happened in that past that drove the true system to generate this output label. But the transition of this unobservable event is missing in the model. In this case we cannot assume a discrepancy of missing a transition with the observable event σ_{k+1}, because the synchronous product guarantees that if σ_{k+1} is defined at z_k, so it is defined in all the components if σ_{k+1} is common, or it is defined in its specific component if it is private. Thus, hypothesizing a transition from any state in the state estimate set z_k will lead to a non-deterministic automaton. The discrepancy will be attributed to an unobservable event.

Thus, the discrepancy handling block in the algorithm flow chart could be explained as follows:

- IF the term $\eta(z_k, \sigma_{k+1})$ evaluates to an empty state.

  Then the type of discrepancy is ***unexpected observable event*** (Missing a transition with an observable event). This depicted in Figure 4.21.

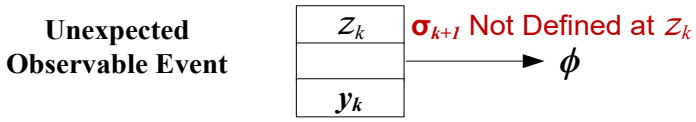


Figure 4.21: Unexpected Observable Event from Case-3

- IF the term $\eta(z_k, \sigma_{k+1})$ evaluates to a non-empty set, but the actual output label cannot be explained where $y_{k+1} \notin \lambda(UR(\eta(z_k, \sigma_{k+1}))$,

  Then the type of discrepancy is ***unexpected output label with observable event*** (Missing a transition with an **un**observable event). This is depicted in Figure 4.22.

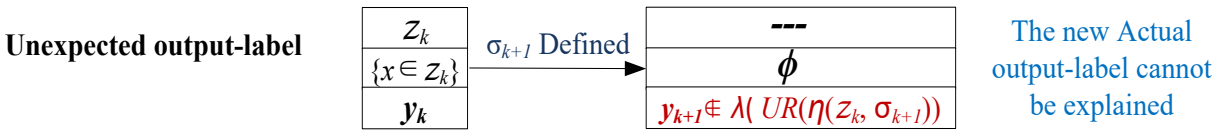


Figure 4.22: Unexpected Output-Label with Observable Event from Case-3

Figure 4.23 explains this logic as a flowchart for Update law of **case-3**.

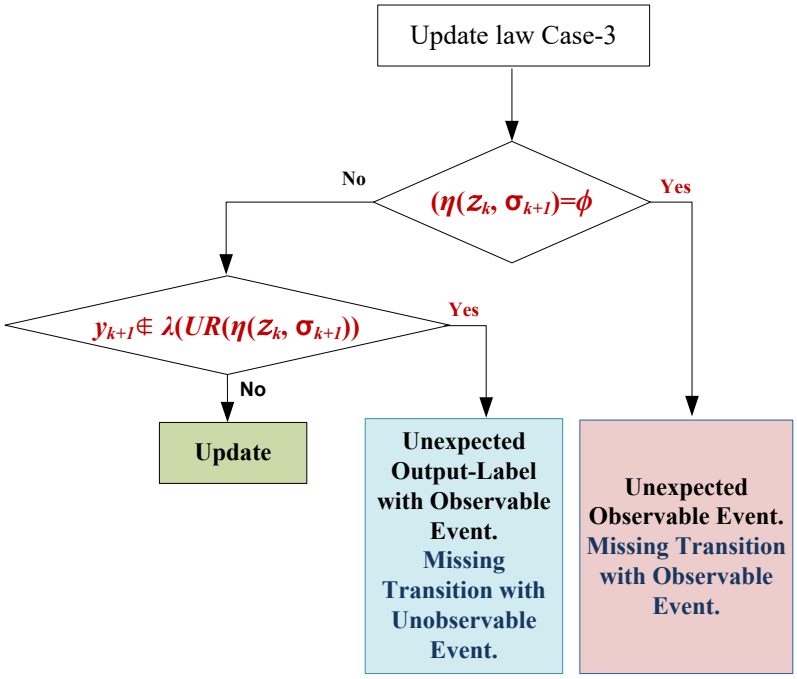


Figure 4.23: Discrepancy Detection by Case-3 Update Law

4.5 Mathematical Formulation for the Learning Algorithm

As explained before, there are two types of missing transitions, the first one is the transitions with observable events while the second is transitions with unobservable events. Therefore, the learning algorithm has two sub-algorithms, the first handles missing transitions with observable events and is denoted as Algorithm-1. The second sub-algorithm denoted as Algorithm-2 handles missing transitions with unobservable events. This section discusses the mathematical formulation of the learning algorithm for each type.

4.5.1 Introduction

By considering X as the set of states of the system, let us assume that it is divided into two subsets. The first subset contains the normal states X_N that are reachable by observable events while the second X_F has all the states reachable by unobservable events where, $X = X_N \cup X_F$. typically, X_F represents the set of faults states, but since some fault events could be observable when measured directly by sensors, then X_F will be denoted in this research as the ***abnormal*** states set, where these states are reachable only by unobservable events.

The output map that was used formerly in the experiments can be reintroduced and defined as the table listing all the states of the system resulting from the cartesian product of the components along with the corresponding generated output labels. This table can be introduced in matrix representation. The rows indexed by the index "i" ranges from 1 to N, where N is the maximum number of rows while the columns are indexed with the index "c" that ranges from 1 to |C|, where C is the set of components in the plant i.e., $C=\{C_1,C_2,\ldots,C_c\}$; so |C| is the number of components. Each row represents a state of the model and is written in form of a **tuple** a $x_i = (x_{i1},x_{i2},\ldots,x_{ic})$, as shown in Table 4.1.

Table 4.1: Output Map

States \ Components	C_1	C_2	…	C_c	y=output label
x_1	$x_{1,1}$	$x_{1,2}$	…	$x_{1,c}$	y_1
x_2	$\underline{x_{2,1}}$	$x_{2,2}$	…	$x_{2,c}$	y_2
x_N	$x_{N,1}$	$x_{N,2}$	…	$x_{N,c}$	y_N

As shown in the output map, $C_1,C_2,\ldots,C_c$ are the component automata. The symbols $x_{i,C}$ are the substates or the state value of the comprising components. For example, the symbol $x_{2,1}$ (underlined in the table) stands for the value of the component C_1 while the model of the system is at state x_2. As another example, let us assume that the state is a 4-tuple (Po,**Po**,**C**,C2) is state number 3 in the table (x_3), then $x_{3,2}$ is **Po** and, $x_{3,3}$ is **C**. Thus, generally for any component's state value written as $x_{i,C}$, the first index "i" specifies the state in the output map which is x_i, while the second index "c" specifies the state value of the component.

4.5.2 Assumptions

Let us review our assumptions on the DES model.

- The number of components is known and does not miss any component model.
- The number of states in the components are known and correct. For example, a valve or a pump automaton has the correct number of states.
- The event set is known and does not miss any event, i.e., the event set of the nominal model and the true model are the same.
- We may only miss some transitions in the model of components (that leads to discrepancy in the composed model).
- The set of output labels is available and accurate (i.e., nothing is missing). In other words, the output map function of the real system and the nominal system are the same.
- The faults (abnormal states) are permanent, and after a failure occurs, the system will not return to normal mode.

4.5.3 Missing Transitions with Observable Events

This section introduces the mathematical formulation in case of the discrepancy occurs following the occurrence of an observable event. By following the same flowchart of the algorithm in section 4.2 , each step of this flowchart is described in the following.

4.5.3.1 Algorithm Description

This section explains in detail the steps of the branch of the learning algorithm concerned with missing transitions with observable events.

1- STEP1: Detect the discrepancy and determine the discrepancy type.

a. <u>Detect discrepancy and its type.</u>

i. Is a discrepancy detected during observer progression?

Answer: Yes

ii. What is the type of discrepancy?

Answer: The new observable event occurred in the system is not defined at the current state estimation of the nominal model, i.e., ***Unexpected Observable Event***.

Cause: Missing transition with observable event.

b. <u>Collect the current system observations and information.</u>

i. Event of Discrepancy σ_d: is the event that happened in the real system that is not defined at current state estimation of the nominal model.

ii. Last Conforming State Set **LCS**:

Based on **ObRule-1**, the last conforming state z_L is the last state estimate before the occurrence of the event of discrepancy as shown in Figure 4.24.

Figure 4.24: Last Conforming State

In Figure 4.24, the last conforming state estimate in the observer of the nominal model is the last state to which the observer moved before the discrepancy (after which the discrepancy event σ_d is **not** defined in the nominal model that results in ϕ estimates).

iii. Actual Label of Discrepancy **ALD:** y_d, is the actual generated label from the real system observed at the discrepancy state.

2- STEP-2: Analyzing the collected information and determining sources and candidates

a. Get Suspected Components

i. Suspected Components SC**:** is the set of components that have the discrepancy event σ_d in their event sets. Since Σ_{Ci} denotes the alphabet of the component C_i then, $\boldsymbol{SC}=\{C_i \mid \sigma_d \in \Sigma_{Ci}, 1 \leq i \leq |C|\}$.

ii. Complete Suspected Components CSC:

This is the subset of suspected components for which σ_d is defined at the current state, i.e., they are the components having σ_d in their event set but found **not missing the transition of σ_d** in at least one of the states of discrepancy. These are the components that are removed from the SC since they are not causing of the discrepancy.

- *The formulation*:

For each estate estimate x_{Lj} in the last conforming state z_L the complete suspected components could be computed by the following formula:

$\boldsymbol{CSC_j} = \{C \in SC \mid \eta(x_{Lj,C}, \sigma_d)\ \textit{is defined}\}\ , \forall\ x_{Lj} \in z_L.$

The index j is used for indexing the states x_{Lj} in z_L where, $1 \leq j \leq |z_L|$.

iii. Deficient Suspected Components DSC

This is the subset of suspected components found with σ_d *NOT* defined at their current state i.e., suspected components found missing a transition with σ_d at the state of discrepancy.

- *The formulation*:

For each estate estimate x_{Lj} in the last conforming state z_L the deficient suspected components could be computed by the following formula:

$\boldsymbol{DSC_j} = \{C \in SC \mid \eta(x_{Lj,C},\ \sigma_d)\ \textit{is not defined}\}\ , \forall\ x_{Lj} \in z_L$

iv. Irrelevant Components Cirv:

Is the set of components that are not suspected of missing the transition with σ_d event at the discrepancy state because they do not have this event in their event sets.

- *The formulation*:

For each estate estimate x_{Lj} in the last conforming state z_L, the irrelevant components could be computed by the following formula:

$\boldsymbol{Cirv} = C - SC$

v. The Deficient Components Table:

It will be required to tabulate the above information so that it can be referenced and indexed in the algorithm. Thus, they will be collected on a table called the Deficient Components Table. To explain how this table is structured, we will introduce the following notations.

Notation 4.1

Consider a certain state x_z,

The deficient part of a state is the values of the deficient components in this state denoted by $x_{z,\{DSC\}}$.

The irrelevant part of a state the values of the irrelevant components in this state denoted by $x_{z,\{\text{Cirv}\}}$.

Example: Assuming a certain state x_z= (C,**Po**,Po,**O**,H) where *DSC*={C2,C4}, then $x_{z,\{DSC\}}$ points to (C2:Po,C4:O) while $x_{z,\{Cirv\}}$ refers to the other components value (C1:C,C3:Po,C5:H).

The *Complete Suspected Components "*CSC" and the *Deficient Suspected Components* "DSC" and

the Irrelevant Components "Cirv" will be used to construct the Deficient Components Table. This is shown by the following example.

Example 4.4: Deficient Components Table

This example is given just to demonstrate constructing the tables required for computations with indexing and filling the cells. An application example will be given after this section of the formulation. Let us assume that we have a plant comprised of four cascaded valves V1,V2,V3,V4 and that an observer detects a discrepancy at a state estimate z_L = {(O,O,Po,O), (O,Po,O,Po), (SO,Po,O,Po)} after an event σ_d occurred. Then, assuming that the *DSC* and *CSC* and *Cirv* are found per each component state estimate x_{Lj}.

The Deficient Components Table could be constructed as follows:

Table 4.2: Deficient Components Table

$x_{Lj} \in z_L$	DSC_j	$x_{Lj,\{DSC\}}$	CSC_j	$x_{Lj,\{CSC\}}$	$Cirv$	$x_{Lj,\{Cirv\}}$
x_{L1}= (O,O,Po,O)	DSC_1= {V2,V3}	$x_{L1,\{DSC\}}$=(V2:O,V3:Po)	CSC_1= {V1}	$x_{L1,\{CSC\}}$=(V1:O)	$Cirv_1$={V4}	$x_{L1,\{Cirv\}}$= (V4:O)
x_{L2}= (O,Po,O,Po)	DSC_2= {V3}	$x_{L2,\{DSC\}}$=V3:Po	CSC_2={V1, V2}	$x_{L1,\{CSC\}}$=(V1:O,V2:Po)	$Cirv_2$={V4}	$x_{L2,\{Cirv\}}$= (V4:Po)
x_{L3}= (SO,Po,O,Po)	DSC_3= {V2,V3}	$x_{L1,\{DSC\}}$=(V2:Po,V3:O)	CSC_3={V1}	$x_{L3,\{CSC\}}$=(V1:Po)	$Cirv_3$={V4}	$x_{L2,\{Cirv\}}$= (V4:Po)

The Deficient Components Table checks each state x_{Lj} in the state estimate z_L and tabulates the corresponding deficient, complete and irrelevant components.

b. Determine the Candidate Suspected States

i. Proposed States X_{PS}

As shown in the experiments and based on **ObRule-3**, it is required to select some states using the output map as suitable candidate states for the destination of the missing transition. So, X_{PS} is the set of states in the output map that have the same label as the *ALD*. In the experiments, these are the states that were first filtered from the output map table.

- *The formulation*:

  $X_{PS}=\{x \mid \lambda(x)=ALD\}$.

Hence, these proposed states are the first group of states proposed by the output map that can generate the label ALD. For example, in a cascaded couple of valves (C,C), (C,O), (O,C), (SC,C), (SC,O) could be the X_{PS} for the label fL. Up to here and based on **ObRule-4**, in terms of selecting and filtering the X_{PS} to get ***candidate expected states***, we have two sub-scenarios for the missing observable event: the first,

the event is private and the second, the event is a common event.

ii. Private Events candidate expected states:

In case of missing transitions with a private observable event, this event belongs to only one automaton and is not common with others, i.e., $|DCS|=1$ or the number of deficient components is only one. In this case it will be required to have some rules to filter the proposed states X_{PS} to get the candidate expected states. These rules are formulated in the following.

- *The formulation*:

If at a certain state estimate z that became the *LCS*, and we have *j* state estimates in this state z such that $z_L=\{x_{L1},x_{L2}, \ldots, x_{Lj}\}$, then the ***candidate expected states*** of each state x_{Lj} could be filtered by applying the following function $X_{cndd}(x_{Lj})$:

$$X_{cndd}(x_{Lj}) = \{x_{PS} \in X_{PS}\}$$

Such that the following conditions hold if applicable:

Table 4.3: Filtering Candidate Expected States – Private Events

	Condition	**Description**
1)	$x_{PS,\{Cirv\}} = x_{Lj,\{Cirv\}}$	Based on **ObRule-5**, select the states $x_{PS} \in X_{PS}$ where the irrelevant part stays unchanged by σ_d.
2)	$x_{PS,\{DSC\}} \notin X_{CF}$ if $x_{Lj,\{DSC\}} \notin X_{CF}$	Based on **ObRule-7**, if the state value of the deficient component at the discrepancy state is normal, select the candidate states to be also normal. Here, X_{CF} is the set of faulty (abnormal) states in the component *C*.
3)	$x_{PS,\{DSC\}} = x_{Lj,\{DSC\}}$ if $x_{Lj,\{DSC\}} \in X_{CF}$	As a result of **ObRule-7**, if the state value of the deficient component at the discrepancy state is abnormal, select the candidate states to be also abnormal and to be the same as the deficient component state value.

By looking to the previous definition and the condition table, we can explain the conditions of selecting the ***candidate expected states*** of each state x_{Lj} as follows:

1) While selecting candidate expected states from X_{PS} ,the irrelevant components' part is held constant as they do not have σ_d in their alphabet and not expecting any change to happen in the irrelevant components.
2) If at a discrepancy state $x_{Lj} \in z_L$, the suspected component is at a normal state such as O or C in a valve, the observable event σ_d should move it to another normal state. Thus, the candidate state inside the component must be also normal. For example, missing ES between ON and OFF in a pump.
3) If at a discrepancy state $x_{Lj} \in z_L$, the suspected component is at a faulty (abnormal) state such as SO or SC in a valve, the event σ_d should move it to an ***applicable*** abnormal state. Thus, the candidate state inside the component must be also faulty (abnormal). But, since it is assumed that the fault modes (abnormal states) are permanent, the candidate state should be selected the same as the state value of the deficient component in LCS. This is because the missing transition with observable event shall not move the automaton from an abnormal state to another. This case resembles missing to model a self-loop at a faulty state, e.g., missing an open command V1O around stuck-open state SO in a valve.

iii. Common Events Candidate States:

- *The formulation*:

If at a certain state estimate z that became the *LCS*, and we have j state estimates in this state z such that $z_L=\{x_{L1},x_{L2}, \ldots, x_{Lj}\}$, then the ***candidate expected states*** of each state x_{Lj} could be filtered by applying the following function $X_{cndd}(x_{Lj})$:

$X_{cndd}(x_{Lj}) = \{x_{PS} \in X_{PS}\}$

Such that the following conditions hold if applicable:

Table 4.4: Filtering Candidate Expected States – Common Events

1)	$x_{PS,\{Cirv\}} = x_{Lj,\{Cirv\}}$	Based on **ObRule-5**, select the states where the irrelevant part stays unchanged by σ_d
2)	$x_{PS,\{CSC\}} = \eta_c(x_{Lj,\{CSC\}},\sigma_d)$	Based on **ObRule-6**, select the states where the complete suspected components having σ_d defined will make a transition by its own transition function η_c.
3)	$x_{PS,\{DSC\}} \notin X_{CF}$ if $x_{Lj,\{DSC\}} \notin X_{CF}$	Based on **ObRule-7**, if the state value of the deficient component at the discrepancy state is normal, select the candidate states to be also normal. Where, X_{CF} is the set of abnormal states in the component C.
4)	$x_{PS,\{DSC\}} = x_{Lj,\{DSC\}}$ if $x_{Lj,\{DSC\}} \in X_{CF}$	As a result of **ObRule-7**, if the state value of the deficient component at the discrepancy state is abnormal, select the candidate states to be also abnormal and to be the same as the deficient component state value.

By looking to the previous definition and the condition table, we can explain the conditions of selecting the ***candidate expected states*** of each state x_{Lj} as follows:

1) While selecting candidate expected states from X_{PS} ,the irrelevant components' part is held constant as they do not have σ_d in their alphabet and not expecting any change to happen in the irrelevant components.
2) The complete suspected components having σ_d common but defined at the discrepancy state, should progress by its own transition function η_c.
3) If the discrepancy state in a suspected component is a normal state such as O or C in a valve, the event σ_d should move it to another normal state. Thus, the candidate state inside the component must be also normal. For example, missing ES between ON and OFF in a pump.
4) If at the discrepancy state, the suspected component is at an abnormal state such as SO or SC in a valve, the event σ_d should not move the automaton to another state because the fault modes (abnormal states) are assumed to be permanent. Thus, we select the candidate state the same as the state value of the deficient component in LCS. This case resembles missing to model a self-loop at a faulty state, e.g., missing an open command V1O around stuck-open state SO in a valve.

For each x_{Lj} in $z_{L,}$ the candidate expected states will be tabulated in table called the Candidate States Table. For the previous demonstration example of the four cascaded valves and assuming that by consulting the output map table, it was possible to get the candidate suspected states as per each state estimate x_{Lj} in the last conforming state.

The Candidate states table is constructed as following example (with these values here for demonstration):

Table 4.5: Demonstration Example - Candidate States Table

x_{Lj}	$X_{cndd}(x_{Lj})$
x_{L1}	(C,C,C,C),(C,Po,Po,C),(C,Po,O,C),(C,O,O,C)
x_{L2}	(C,C,C,C),(C,C,Po,C),(C,C,O,C)
x_{L3}	(C,C,C,C)

c. Determine the Destinations of the Missing Transitions

Destinations Table:

This table is computed based on Table 4.5 of candidate states table. Each field in the destinations table will contain the proposed or candidate destination state of the component C_i per each x_{Lj}. The destination of a transition will be formulated in the form a colon pair as $C_i{:}x_t$ where C_i is the component name and x_t is the state value of that component at its destination. The formula to get each field in this destinations table is as follows.

- *The formulation*:

For a component C_i indexed with "i" in a candidate state $x_{cndd\_k}(x_{Lj})$ indexed with "k" for a state estimate x_{Lj} in the last conforming state z_L, the destinations could be represented by the following formula:

$$Dest_{j,k,i} = Ci{:}\ x_{cndd\_k,i}(x_{Lj}) \text{ if } Ci \in DSC_j \forall\ 1 \le j \le |z_L|, \forall\ 1 \le k \le |X_{cndd}(x_{Lj})|, \forall\ 1 \le i \le |C|$$

This means for each state estimate x_{Lj} in LCS, there will be some candidate states $X_{cndd}(x_{Lj})=\{x_{cndd\_1}(x_{Lj}), \dots, x_{cndd\_k}(x_{Lj})\}$. In these candidate states, the value of the component C_i is the destination of the transition if this C_i belongs to the deficient components DSC_j of this candidate state $x_{cndd\_k}(x_{Lj})$.

For the previous demonstration example, and based on the Deficient Components Table, the Destination

Table will be constructed as follows:

Table 4.6: Demonstration Example - Destination Table

x_{Lj}	$x_{cndd\_k}(x_{Lj})$	$Dest_{j,k,i} = Ci: x_{cndd\_k,i}(x_{Lj})$			
		V1	**V2**	**V3**	**V4**
x_{L1}=(O,O,Po,O)	$x_{cndd\_1}(x_{L1})$=(C,C,C,C)	---	V2:C	V3:C	---
	$x_{cndd\_2}(x_{L1})$=(C,Po,Po,C)	---	V2:Po	V3:Po	---
	$x_{cndd\_3}(x_{L1})$=(C,Po,O,C)	---	V2:Po	V3:O	---
	$x_{cndd\_4}(x_{L1})$=(C,O,O,C)	---	V2:O	V3:O	---
x_{L2}=(O,Po,O,Po)	$x_{cndd\_1}(x_{L2})$=(C,C,C,C)	---	---	V3:C	---
	$x_{cndd\_2}(x_{L2})$=(C,C,Po,C)	---	---	V3:C	---
	$x_{cndd\_3}(x_{L2})$=(C,C,O,C)	---	---	V3:C	---
x_{L3}=(SO,Po,O,Po)	$x_{cndd\_1}(x_{L3})$=(C,C,C,C)	---	V2:C	V3:C	---

d. Determine the Sources of Missing Transitions

Sources Table:

Based on **ObRule-2**, the source of missing transitions in the suspected components are written in the form of colon pairs as Ci:$x_{Lj,i}$. The first element in the colon pair is the Deficient Suspected Component and the second element is the state value of this component in the last conforming state estimate x_{Lj}.

To backwardly link the destinations table to the sources, the source of missing transition is computed for each deficient component cell in the Destinations Table. This associates every cell in the destination table to its corresponding originating source to facilitate generating the hypotheses. This is depicted in Figure 4.25.

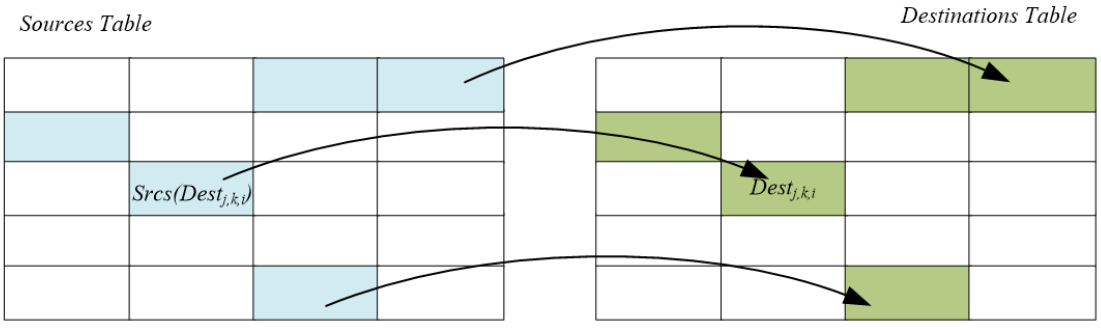


Figure 4.25: Sources Table to Destination Tables Correspondence

The source of missing transition is computed for each deficient component cell in the Destinations Table $Dest_{j,k,i}$ using the following formula.

- *The formulation*:

For a component C_i indexed with "i" in a candidate state $x_{cndd\_k}(x_{Lj})$ indexed with "k" for a state estimate x_{Lj} indexed with "j" in the last conforming state z_L, the sources could be given by the following function:

$$Srcs(Dest_{j,k,i}) = C_i{:}x_{Lj,i} \quad \text{if } C_i \in DSC_j \quad \forall\ 1 \le j \le |z_L|,\ \forall\ 1 \le k \le |X_{cndd}(x_{Lj})|,\ \forall\ 1 \le i \le |C|$$

For the previous example the sources table will be constructed as follows:

Table 4.7: Demonstration Example - Sources Table

x_{Lj}	$x_{cndd\_k}(x_{Lj})$	V1	V2	V3	V4
x_{Lj1}=(O,O,Po,O)	$x_{cndd\_1}(x_{L1})$=(C,C,C,C)	---	V2:O	V3:Po	---
	$x_{cndd\_2}(x_{L1})$=(C,Po,Po,C)	---	V2:O	V3:Po	---
	$x_{cndd\_3}(x_{L1})$=(C,Po,O,C)	---	V2:O	V3:Po	---
	$x_{cndd\_4}(x_{L1})$=(C,O,O,C)	---	V2:O	V3:Po	---
x_{L2}=(O,Po,O,Po)	$x_{cndd\_1}(x_{L2})$=(C,C,C,C)	---	---	V3:O	---
	$x_{cndd\_2}(x_{L2})$=(C,C,Po,C)	---	---	V3:O	---
	$x_{cndd\_3}(x_{L2})$=(C,C,O,C)	---	---	V3:O	---
x_{L3}=(SO,Po,O,Po)	$x_{cndd\_1}(x_{L3})$=(C,C,C,C)	---	V2:Po	V3:O	---

3- STEP-3: Hypotheses generation and model compliance verification

This step is based on **ObRule-8** to **11**.

a. Generating the Hypotheses

i. Hypotheses

After computing the sources and destinations tables, it is possible to interlink between the corresponding cells in the tables to generate the hypotheses. As mentioned before such tabulation will be beneficial in organizing the work to get it programmatically implemented through programming loops.

Based on **ObRule-8**, the hypotheses are computed for each deficient component C_i per each candidate

state of each last conforming state x_{Lj}. Each hypothesis starts from the source of the missing transition to the candidate destination with the event of discrepancy σ_d.

ii. <u>The simple hypothesis</u>

The hypothesis is simple if the missing transition is proposed in single component, and it starts from the source to the destination by the event of discrepancy in the component itself.

- *The formulation*:

$$H_{j,k,i} = Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_d} Dest_{j,k,i} \; \forall \; 1 \leq j \leq |Z_L|, \; \forall \; 1 \leq k \leq |X_{cndd}(x_{Lj})|, \; \forall \; 1 \leq i \leq |C|$$

In the previous example, the simple hypothesis of x_{L2} at the first candidate of x_{L2} and the third component V3 is:

$$H_{2,1,3} = V3{:}O \xrightarrow{\sigma_d} C$$

Sources Table

x_{Lj}	$x_{cndd\_k}(x_{Lj})$	V1	V2	V3	V4
x_{L1}=(O,O,Po,O)	$x_{cndd\_1}(x_{L1})$=(C,C,C,C)	---	V2:O	V3:Po	---
	$x_{cndd\_2}(x_{L1})$=(C,Po,Po,C)	---	V2:O	V3:Po	---
	$x_{cndd\_3}(x_{L1})$=(C,Po,O,C)	---	V2:O	V3:Po	---
	$x_{cndd\_4}(x_{L1})$=(C,O,O,C)	---	V2:O	V3:Po	---
x_{L2}=(O,Po,O,Po)	$x_{cndd\_1}(x_{L2})$=(C,C,C,C)	---	---	V3:O	---
	$x_{cndd\_2}(x_{L2})$=(C,C,Po,C)	---	---	V3:O	---
	$x_{cndd\_3}(x_{L2})$=(C,C,O,C)	---	---	V3:O	---
x_{L3}=(SO,Po,O,Po)	$x_{cndd\_1}(x_{L3})$=(C,C,C,C)	---	V2:Po	V3:O	---

Destinations Table

V1	V2	V3	V4
---	V2:C	V3:C	---
---	V2:Po	V3:Po	---
---	V2:Po	V3:O	---
---	V2:O	V3:O	---
---	---	V3:C	---
---	---	V3:C	---
---	---	V3:C	---
---	V2:C	V3:C	---

Figure 4.26: Demonstration Example – Simple Hypothesis Sources to Destination Linking

iii. <u>Combined Hypothesis:</u>

A hypothesis might be comprised of more than one transition at a time if the event of discrepancy is common and is found missing in more than one component. Thus, for each state estimate, the hypothesis is generated as the combination of the simple transition hypotheses ending by the corresponding *candidate expected states*. For the previous example:

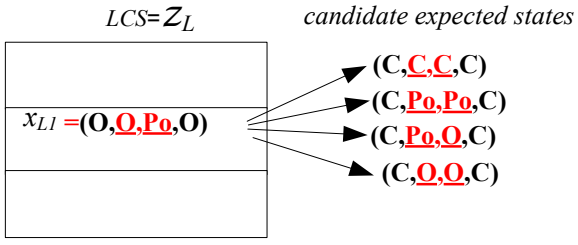


Figure 4.27: Demonstration Example - Hypotheses Generation

The ***candidate expected state*** for x_{L1} denoted by $X_{cndd}(x_{L1})$ = {(C,C,C,C), (C,Po,Po,C), (C,Po,O,C), (C,O,O,C)} = $\{X_{cndd\_1}(x_{L1}), X_{cndd\_2}(x_{L1}), X_{cndd\_3}(x_{L1})\}$

The hypotheses will be generated respecting the combination as follows:

$H_1(x_{L1}) = V2{:}O \overset{\sigma_d}{\rightarrow} C$ AND $V3{:}Po \overset{\sigma_d}{\rightarrow} C$

$H_2(x_{L1}) = V2{:}O \overset{\sigma_d}{\rightarrow} Po$ AND $V3{:}Po \overset{\sigma_d}{\rightarrow} Po$

$H_3(x_{L1}) = V2{:}O \overset{\sigma_d}{\rightarrow} Po$ AND $V3{:}Po \overset{\sigma_d}{\rightarrow} O$

$H_4(x_{L1}) = V2{:}O \overset{\sigma_d}{\rightarrow} O$ AND $V3{:}Po \overset{\sigma_d}{\rightarrow} O$

Thus, if we have *K candidate expected states,* where "*k"* is used to index them, then the combined hypothesis could be formulated as follows:

- *The formulation*:

$H_{j,k}$ = AND $[Srcs(Dest_{j,k,i}) \overset{\sigma_d}{\rightarrow} Dest_{j,k,i}]$ $\forall\ 1 \leq j \leq |Z_L|$, $\forall\ 1 \leq k \leq |X_{cndd}(x_{Lj})|$, $\forall\ 1 \leq i \leq |C|$

Note that, the index "*i*" in hypothesis name is removed because the hypothesis now is combined of multiple components hypotheses.

b. Synchronizing the Corrected Components and Generate New Models

The previous formulas may result in a set of hypotheses for the corrected components. Consequently, the corrected components shall undergo a new synchronization operation that generates a new set of hypothesized models. This set of models have their progression monitored by the observer. If any of the new models becomes incompliant with the true plant observations while there are other models complying with the true plant, the incompliant model shall be rejected. This rejection is based on the fact that there are some models having explanation for the plant data while the rejected model perhaps was built using incorrect hypothesis and it is the time to reject it.

- *The formulation*:

If a number of hypotheses *M* indexed by *m* are generated and the number of components is |*C*| thus, the hypothesized models are given by:

$Model_m$= Synch ($C_1, C_2, \ldots, C_c$), $1 \leq m \leq M$.

4.5.3.2 Observable Events Algorithm pseudo code

The following pseudo code represents the learning algorithm for the missing transitions with an

observable event. The code starts from detecting a discrepancy to generating the hypothesized models. It should be noted, the text starting with sign "/*" is considered as an informational comment line.

Algorithm 1: Observable Events Learning Algorithm

Input: *Current nominal model,* $\boldsymbol{\sigma_d}$ *,**ALD**, **LCS**.*

Output: *Set of hypothesized models*.

1: Suspected Components $\boldsymbol{SC} = \{C \mid \sigma_d \in \Sigma_c\}$

2: **For each** $x_{Lj} \in z_L$

3: Complete Suspected Components $\boldsymbol{CSC_j} = \{C \in SC \mid \eta(x_{Lj,C}, \sigma_d)$ **IS DEFINED**$\}$

4: Deficient Suspected Components $\boldsymbol{DSC_j} = \{C \in SC \mid \eta(x_{Lj,C}, \sigma_d)$ **IS NOT DEFINED**$\}$

5: Irrelevant Components $\boldsymbol{Cirv_j} = C - SC$

6: Fill in The Deficient Components Table

7: **End**

8: /* Determine the Candidate suspected states

9: Proposed States $X_{PS}=\{x \mid \lambda(x)=ALD\}$

10: **For each** $x_{Lj} \in z_L$, SELECT FROM X_{PS} WHERE

11: */* Candidate expected states (with Private event or Common event)*

$X_{cndd}(x_{Lj}) = \{x_{PS} \in X_{PS}\}$

12: AND $x_{PS,\{Cirv\}} = x_{Lj,\{Cirv\}}$

13: AND $x_{PS,\{CSC\}} = \eta_c(x_{L,\{CSC\}}, \sigma_d)$

14: AND **IF** $x_{Lj,\{DSC\}} \notin X_{CF}$

15: **THEN** $x_{PS,\{DSC\}} \notin X_{CF}$

16: AND **IF** $x_{Lj,\{DSC\}} \in X_{CF}$

17: **THEN** $x_{PS,\{DSC\}} = x_{Lj,\{DSC\}}$

18: Fill in The Candidate states table

19: **End**

20: /* Destinations Table

21: **For each** $x_{Lj} \in z_L$

22: **For each** $x_{cndd\_k}(x_{Lj})$ in $X_{cndd\_k}(x_{Lj})$

23: **For each** C_i in $x_{cndd\_k}(x_{Lj})$

24: $Dest_{j,k,i} = Ci: x_{cndd\_k,i}(x_{Lj})$ if $Ci \in DSC_j$

25: Fill Destinations Table with $Dest_{j,k,i}$

26: **End**

27: **End**

28: **End**

29: /* Sources Table

30: **For each** $x_{Lj} \in z_L$

31: **For each** $x_{cndd\_k}(x_{Lj})$ in $X_{cndd\_k}(x_{Lj})$

32: **For each** C_i in $x_{cndd\_k}(x_{Lj})$

33: $Srcs(Dest_{j,k,i}) = C_i: x_{Lj,i}$ IF $C_i \in DSC_j$

34: Fill in Sources Table with $Srcs(Dest_{j,k,i})$

35: **End**

36: **End**

36: **End**

37: /* Generate the Hypotheses

38: **For each** $x_{Lj} \in z_L$

39: **For each** $x_{cndd\_k}(x_{Lj})$ in $X_{cndd\_k}(x_{Lj})$

40: **For each** C_i in $x_{cndd\_k}(x_{Lj})$

41: $H_{j,k,i} = Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_d} Dest_{j,k,i}$

42: **End**

43: **End**

44: **End**

45: /* combined Hypotheses

46: $H_{j,k}$ = AND $[Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_d} Dest_{j,k,i}] \; \forall \; 1 \le j \le |z_L|, 1 \le k \le |Xcndd(x_{Lj})|, 1 \le i \le |C|$

47: /* Synchronize the corrected components and generate the new models

48: **For each** H in $H_{j,k,i}$

49: $Model_m$= Synch $(C_n)_m$, $1 \le n \le |C|$, $1 \le m \le M$, M is the number of hypotheses

50: **End**

4.5.3.3 Application Example-1

The system in this example is comprised of cascaded pump and valve as shown in Figure 4.28. The valve V1 is similar to the examples given before in terms of the states and events. The pump has two states Off and On. The events in the pumps are:

p2Stp: Stop command **p2Rn**: Run command **ES**: Emergency Shut off

The model is built while missing to model ES between On and Off in P2, while this event exists and can occur in the real plant if the pump was On. This example will go through the progression of the model until detecting the discrepancy then applying the learning algorithm to correct it by generating hypotheses.

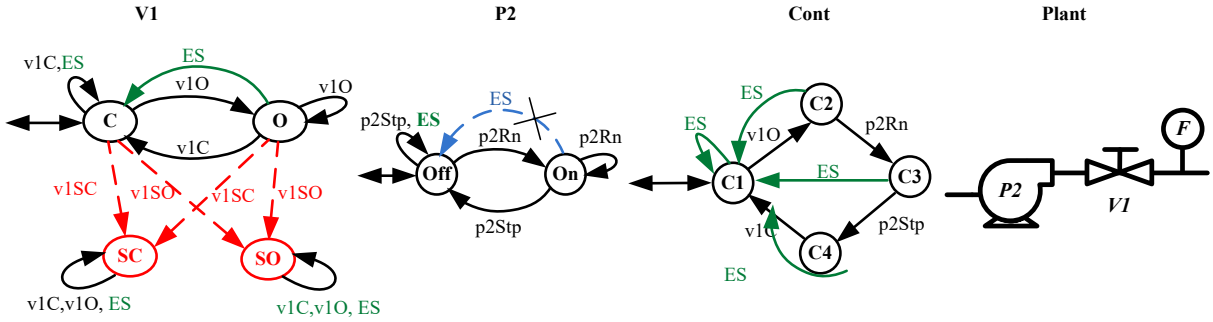


Figure 4.28: Observable Events Application Example-1

Figure 4.29 shows the automaton of the nominal model after the synchronous product of the components as G = synch (V1,P2,Cont).

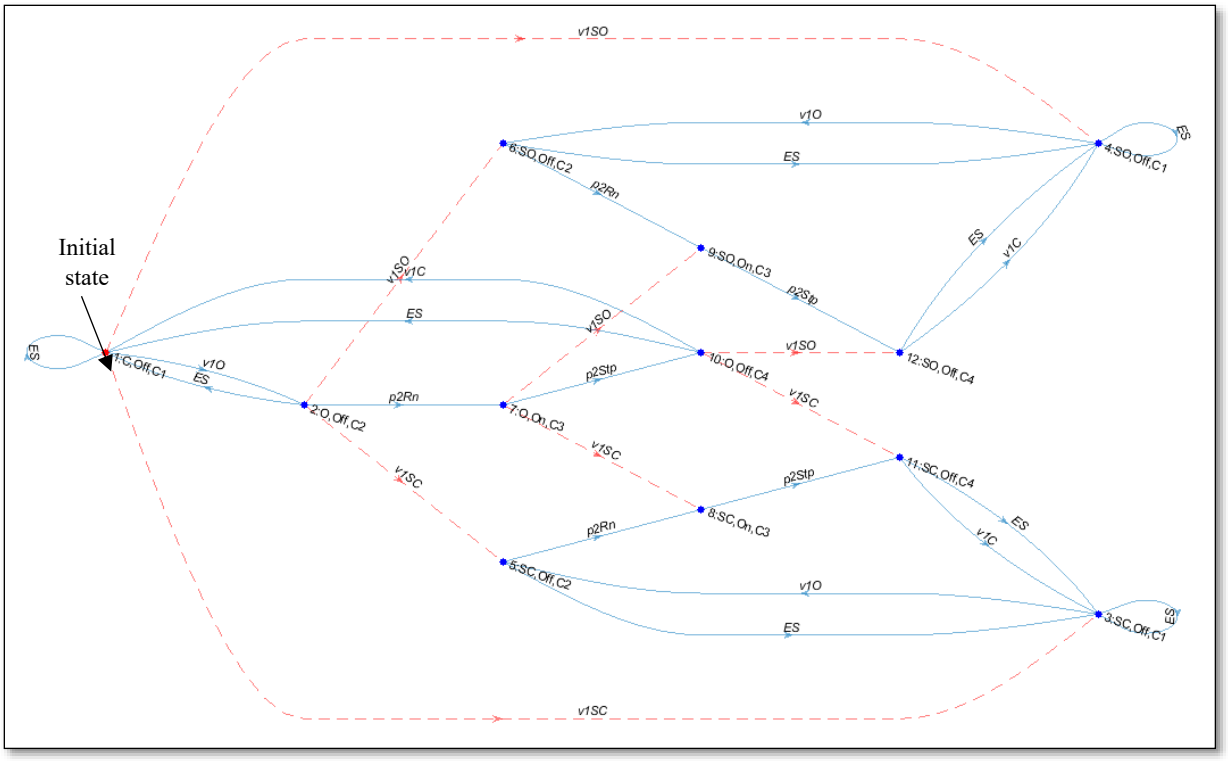


Figure 4.29: Example-1 Nominal Model

The system was initialized, and the model starts with initial state z_0

Since $z_0 = \{x \mid x \in UR(\{x_0\})$ and $\lambda(x) = \lambda(x_0)\}$

$z_0 = \{1,3,4\}$

y_0 = fL

➔ **V1O , y_1=fL:** new event occurred at the real system with no label change (**Case-1**)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$

Then, $z_1 = \{x \mid x \in UR(\eta(z_0, \mathbf{V1O}))$ and $\lambda(x)=\mathbf{fL}\}$.

Since $z_0 = \{1,3,4\}$, $\eta(z_0, \mathbf{V1O}) = \{2,5,6\}$ as explained below.

$x \in z_0$	$\eta(x,$**V1O** $)$	$UR(.)$	UR and $\lambda(x)$=fL
1	$\eta(1,$**V1O**$)=2$	{2,5,6}	{2,5,6}
3	$\eta(3,$**V1O**$)=5$	{5}	{5}
4	$\eta(4,$**V1O**$)=6$	{6}	{6}

Thus, $z_1 = \{2,5,6\}$.

➔ **P2On , y_2= fH:** new event occurred with label change (Case-3)

Since $z_{k+1} = \xi_e(z_k, \sigma_{k+1}, y_{k+1})$, $\xi_e(z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_{k+1}\}$

Then, $z_2 = \{x \mid x \in UR(\eta(z_1, \mathbf{P2On}))$ and $\lambda(x)=\mathbf{fH}\}$

Since $z_1 = \{2,5,6\}$, $\eta(z_1, \mathbf{P2On}) = \{7,8,9\}$ as explained below.

$x \in z_1$	$\eta(x,\text{P2O})$	$UR(.)$	UR and $\lambda(x)=\mathbf{fH}$
2	$\eta(2,\mathbf{P2On})=7$	{7,8,9}	{7,9}
5	$\eta(5,\mathbf{P2On})=8$	{8}	---
6	$\eta(6,\mathbf{P2On})=9$	{9}	{9}

But $\lambda(8) \neq \mathbf{fH}$, thus, $z_2 = \{7,9\}$.

➔ ES, $y_3 = \mathbf{fL}$: new event occurred with label change (Case-3)

$z_3 = \{x \mid x \in UR(\eta(z_2, \mathbf{ES}))$ and $\lambda(x)=\mathbf{fL}\}$

Since $z_2 = \{7,9\}$, $\eta(z_2, \mathbf{ES}) = \phi$ as explained below.

$x \in z_2$	$\eta(x \in z_2)$	$UR(.)$	$UR \mid \lambda(x)=\mathbf{fL}$
7	$\eta(7,\mathbf{ES})$= NOT DEFINED	---	---
9	$\eta(9,\mathbf{ES})$= NOT DEFINED	---	---

ES is **NOT DEFINED** at $z_2 = \{7,9\}$, **DISCREPANCY DETECTED**. The progression of the observer during the previous steps is depicted in Figure 4.30.

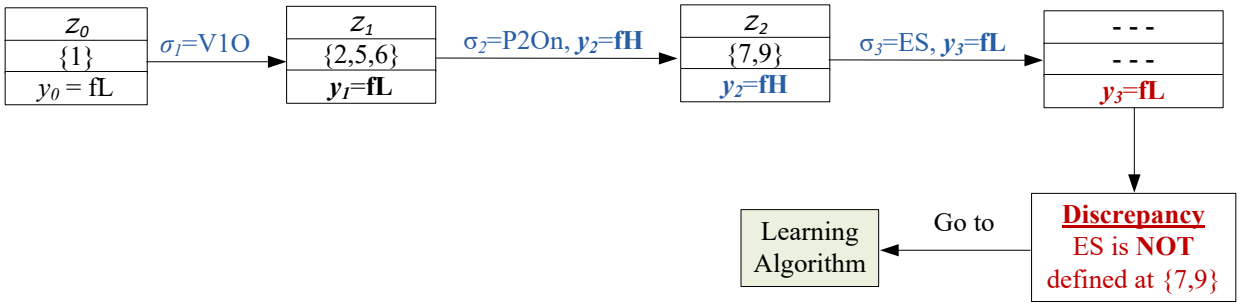


Figure 4.30: Example-1 Model Progression

1- STEP1: Detect the discrepancy and determine the discrepancy type.

a. Detect discrepancy and its Type

i. Type of Discrepancy: Unexpected observable event.

ii.Cause: Missing Transition with Observable Event

b. Collect the current system observations and information

i. Event of discrepancy $\sigma_d = \mathbf{ES}$

ii. Actual Label of Discrepancy **ALD**= y_d= **fL**

iii. Last Conforming State **LCS:** $z_L = z_2 = \{7,9\} = \{(O,On,C3),(SO,On,C3)\} = \{x_{L1}, x_{L2}\}$

2- STEP-2: Analyzing the collected information and concluding sources and candidates

a. Suspected Components:

Since $\boldsymbol{SC} = \{C \mid \sigma_d \in \Sigma_c\}$ and σ_d=ES, and ES $\in$ {V1, P2,Cont}

The SC = {V1,P2,Cont}

At x_{L1}= (O,On,C3)

ES is defined at V1:O

ES is **NOT** defined at P2:On

ES is defined at Cont:C3

Then, $\mathbf{DSC_1}$ ={P2} (Deficient Suspected Components)

CSC_1={V1,Cont} (Complete Suspected Components)

Cirv= --- (Irrelevant Components)

At x_{L2}= (SO,On,C3)

ES is defined at V1:SO

ES is **NOT defined** at P2:On

ES is defined at Cont:C3

Then, $\mathbf{DSC_2}$ ={P2}

CSC_2={V1,Cont}

Cirv= ---

Deficient Components Table

$x_{Lj} \in z_L$	DSC_j	$x_{Lj,\{DSC\}}$	CSC_j	$x_{Lj,\{CSC\}}$	$Cirv$	$x_{Lj,\{Cirv\}}$
x_{L1}=(O,On,C3)	DSC_1={P2}	$x_{L1,\{DSC\}}$=(P2:On)	CSC_1={V1,Cont}	$x_{L1,\{CSC\}}$=(V1:O, Cont :C3)	---	---
x_{L2}=(SO,On,C3)	DSC_2={P2}	$x_{L2,\{DSC\}}$=(P2:On)	CSC_2={V1,Cont}	$x_{L2,\{CSC\}}$=(V1:O, Cont :C3)	---	---

Up to here, we can omit the values of the controller Cont., because it is assumed to be known by the designer and it is not missing the event while only physical components are used to build the output map.

b. Determine the Candidate Suspected States

Proposed States X_{PS}

$X_{PS}=\{x \mid \lambda(x)=ALD\}$, Thus $X_{PS}=\{x \mid \lambda(x)=\text{fL}\}$

From the output map, the proposed states having output label = fL will be selected.

V1	P2	Output
C	On	fL
C	Off	fL
SC	On	fL
SC	Off	fL
O	Off	fL
SO	Off	fL

Candidate expected states

x_{Lj}	$X_{cndd}(x_{Lj})$	Description
x_{L1}=(O,On)	(C,On), (C,Off)	Complete suspected component V1 moved from O to C by the common event ES in the expected candidate.
x_{L2}=(SO,On)	(SO,Off)	Complete suspected component V1 is at an abnormal state SO, so it should not change in the expected candidate.

c. Determine the Destinations of the Missing Transitions

Destinations Table:

$Dest_{j,k,i} = Ci: x_{cndd\_k,i}(x_{Lj})$ if $Ci \in DSC_{j,k} \forall\ 1 \le j \le |\mathcal{Z}_L=LCS|, \forall\ 1 \le k \le |X_{cndd}(x_{Lj})|, \forall\ 1 \le i \le |C|$

x_{Lj}	$x_{cndd\_k}(x_{Lj})$	V1	P2
x_{L1}=(O,On)	$x_{cndd\_1}(x_{L1})$=(C,**On**)	---	P2:On
	$x_{cndd\_2}(x_{L1})$=(C,**Off**)	---	P2:Off
x_{L2}=(SO,On)	$x_{cndd\_1}(x_{L2})$=(SO,**Off**)	---	P2:Off

d. Determine the Sources of Missing Transitions

Sources Table

$Srcs(Dest_{j,k,i}) = C_i:x_{Lj,i}$ if $C_i \in DSC_{j,k}$ $\forall\ 1 \le j \le |\mathcal{Z}_L=LCS|, \forall\ 1 \le k \le |Xcndd(x_{Lj})|, \forall\ 1 \le i \le |C|$

x_{Lj}	$x_{cndd\_k}(x_{Lj})$	V1	P2
x_{L1}=(O,**On**)	$x_{cndd\_1}(x_{L1})$=(C,On)	---	P2:On
	$x_{cndd\_2}(x_{L1})$=(C,Off)	---	P2:On
x_{L2}=(SO,**On**)	$x_{cndd\_1}(x_{L2})$=(SO,Off)	---	P2:On

3- STEP-3: Hypotheses generation and model compliance verification

Hypotheses:

$H_{j,k,i} = Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_d} Dest_{j,k,i}\ \forall\ 1 \leq j \leq |z_L = LCS|,\ \forall\ 1 \leq k \leq |X_{cndd}(x_{Lj})|,\ \forall\ 1 \leq i \leq |C|$

$H_{1,1,2} =\ P2{:}\text{On} \xrightarrow{ES} \text{On}$

$H_{1,2,2} =\ P2{:}\text{On} \xrightarrow{ES} \text{Off}$

$H_{2,1,2} =\ P2{:}\text{On} \xrightarrow{ES} \text{On}$ *Repeated*

The control sequence V1,P2Rn,P2Stp,V1C will lead to reject the first hypothesis, since after ES, the model (of $H_{1,1,2}$) will expect that the pump is running and when it will receive the first event V1O, it will expect high flow "fH" while the actual output will be low flow "fL". This output will be complying with the model built with the hypothesis $H_{1,2,2}$.

Thus, in this example, it is possible to hypothesize the missing transition and to reject the incompliant hypothesis.

4.5.4 Missing Transitions with Unobservable Events

This section introduces the mathematical formulation in case of the discrepancy pertains to missing transitions with **unobservable** events. By following the learning algorithm general flowchart, each step of this flowchart is described in the following.

4.5.4.1 Algorithm Description

This section explains in detail the steps of the learning algorithm concerned with missing transitions with unobservable events. In case of a discrepancy due to a missing transition with an unobservable event, there will be two types of the symptoms of discrepancies as follows.

Type-1: Unexpected Output Label with Observable Event: after the occurrence of an observable event, a wrong output is expected by the model, while the unexpected actual output could not be explained.

Type-2: Unexpected Output-Label: The real system changes the output without any observable event occurred where this new output could not be explained by the nominal model.

The algorithm has three main steps or stages. The first is to determine the discrepancy and to collect the

system's data. The second will analyze the data and determine the sources and destinations of the missing transitions. The third is for generating hypotheses. Therefore, STEP-1 of the algorithm that determines the type of discrepancy will discuss each type of discrepancy symptoms separately then the following steps will be the same for both types.

1- STEP1: Detect the discrepancy and determine the discrepancy type.

For Type-1: Unexpected Output Label with Observable Event

a. Detect Discrepancy and its Type

i. Is a discrepancy detected during observer progression?

Answer: Yes

ii. What is the type of discrepancy?

Answer: An observable event occurred that is defined at the last state estimate, but the actual output label from the true system cannot be explained by the model, i.e., unexpected output label with observable event.

Cause: Missing transition with unobservable event. This is based on **UnObRule-1.a**.

b. Collect the current system observations and information

i. The event of discrepancy σ_d: is the observable event that occurred and after which system generated an output label that could not be explained by the model.

ii. Actual Label of Discrepancy **ALD** : is the actual label generated by the true system at the discrepancy state.

In this case, the actual output is mismatched with the expected by the model. This means that the output label generated by the true system couldn't be explained by the current nominal model at the current state estimate of the observer.

iii. Last Conforming State Set **LCS**: z_L is the last state estimated by the observer before the discrepancy state estimate where the output label generated by the true system could be explained by the state estimate of observer.

iv. Discrepancy State **DS**: z_d in this case, the discrepancy state is the set of states estimates that results by the transition from the LCS with the event of discrepancy σ_d. The Discrepancy state is called the *Expected State* in the experiments where it is the state expected by the nominal model after having σ_d (based on **UnObRule-2**). This is depicted in Figure 4.31.

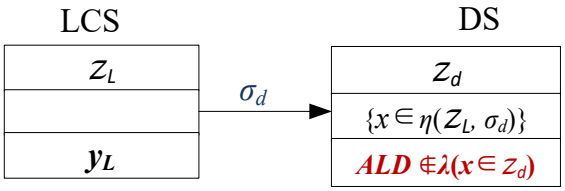


Figure 4.31: Discrepancy State

Note: this variable **DS** was not defined in the algorithm of missing transition with **observable** events, because the progression stops after the occurrence of the event of discrepancy that evaluates to ϕ.

Type-2 Unexpected Output Label.

a. Detect Discrepancy and its Type

i. Is a discrepancy detected during observer progression?

Answer: Yes

ii. What is the type of discrepancy?

A sudden output label change in the true system with no observable event occurred and the output label could not be explained by the model, i.e., unexpected output label.

Cause: Missing transition with unobservable event. This is based on **UnObRule-1.b**.

b. Collect the Current System Observations and Information

i. Actual Label of Discrepancy ALD : y_d is the actual new generated label from the real system observed as discrepancy after the last conforming state estimate.

ii. Last Conforming State set LCS**:** z_L is defined in this case as the last state during the progression of nominal model that can explain the label y_d at the true system. While being in LCS, the real system generates an unexpected new label that has no explanation by the current nominal model and causes discrepancy.

iii. Discrepancy State set DS**:** z_d the discrepancy state estimate here will be taken as the last state estimate reached by the observed nominal model. The Discrepancy state was called the *Expected State* in the experiments (based on **UnObRule-2**). This is because here we do not have an event of discrepancy σ_d.

We can note in this case that the last conforming state estimate equals the discrepancy state DS i.e., **DS = LCS**.

2- STEP-2: Analyzing the Collected Information and Concluding Sources and Candidates

This algorithm has different order of steps than the algorithm of missing transition with observable

events.

a. Determine the Deficient Suspected Components:

First we have to find the suspected states that could be reached by the true model after the unobservable event occurred and will be abbreviated as **SS**.

i. Suspected States **SS**: is the set of abnormal states in the output map table that have the same output label as ALD. This is based on **UnObRule-3**.

- *The formulation*:

SS = $\{x \in X_F \mid \lambda(x) = y_d\}$.

Candidate States: $\boldsymbol{X_{cndd}(x_{dj})}$: Set of states filtered from SS states to be used as the candidates for the expected states instead of the wrongly expected DS by the model (based on **UnObRule-4.a, b and c**). For each $x_{dj} \in DS$, the corresponding candidate states is filtered form the SS states such that they have the maximum similar part of components states as in the x_d. This similar part is denoted as the ***correct expected part*** while the remaining is denoted as the ***wrong expected part***. Since we assumed during the introduction of output map, that we can represent the model states in form of a tuple, then we can segment the tuple of a candidate state as:

$x_{cndd} \in Xcndd$ as: $x_{cndd} = (x_{cndd,\{CRCT\}}, x_{cndd,\{WRNG\}})$.

This means that a candidate state has a correct expected part suffixed with {CRCT} and wrong expected part suffixed with {WRNG}.

Example 4.5:

This example explains the notations of correct and wrong expected parts. Let us assume that we have a plant of 5 cascaded valves, and the discrepancy state estimate reached by the observer say z_d = $\{x_{d1},x_{d2}\}$={(O,Po,O,Po,Po), (O,O,Po,Po,O)}. Assuming the filtered candidate states are $X_{cndd}(x_{d1})$ = {(O,Po,SO,Po,Po), (O,Po,O,Po,SO)} and $X_{cndd}(x_{d2})$ = {(O,SO,Po,Po,O), (O,O,Po,Po,SO)}. This is shown in Table 4.8.

Table 4.8: Example of Wrong Expected Parts

x_{dj}	$X_{cndd}(x_{Li})$	
x_{d1}=(O,Po,O,Po,Po)	(O,Po,SO,Po,Po)	Same output as z_d
	(O,Po,O,Po,SO)	Same output as z_d
x_{d2}=(O,O,Po,Po,O)	(O,SO,Po,Po,O)	Same output as z_d
	(O,O,Po,Po,SO)	Same output as z_d

For the x_{d1}= (O,Po,**O**,Po,Po) having the first candidate destination state of (O,Po,SO,Po,Po), the correct part is {V1,V2,V4,V5} and the wrong part is {V3}.

Thus, the correct part could be written as $x_{d1,\{CRCT\}}=x_{d1,\{1,2,4,5\}}$= (V1:O, V2:Po, V4:Po, V5:Po). Similarly, the wrong part will be written as $x_{d1,\{WRNG\}} = x_{d1,\{3\}}$ = (V3:O).

In fact, the wrong part contains the suspected components because they have been wrongly expected due to an unobservable event with its transition was missing in the model.

Thus, the aim here is to filter the suspected states SS to a narrower list of the candidate states by selecting the group of states with the maximum correct part. This selection could be formulated as follows:

- *The formulation*:

The Candidate States for a state estimate $x_{dj} \in DS$ can be selected from the suspected states SS by applying the following function:

$X_{cndd}(x_{dj}) = \{x_{ss} \in SS \mid x_{ss,\{CRCT\}} = x_{dj,\{CRCT\}}\}$, $\forall\, x_{dj} \in z_d$ Such that the following conditions hold if applicable:

Table 4.9: Filtering Candidate Expected States – Unobservable Events

1)	$\lvert\{CRCT\}\rvert$ is maximum	The cardinality (the length) of the correct part is selected to be maximum, i.e., the states with biggest correct part are selected.
2)	$x_{ss,\{WRNG\}}$ should belong to abnormal states.	This means that the wrong part in the candidate expected states should be comprised of abnormal states of the components. This is because the missing transition we are searching for has unobservable event that ends with an abnormal state.
3)	Abnormal states in components are permanent.	If a component was in an abnormal state at DS then the transition of missing event should not move it to another different state. This means during the selection, we select the states that has the abnormal component value unchanged.

Sources and destinations of the missing transitions will be represented in the form of tables. This means that the sources will be computed to fill in a table called the sources table. Similarly, the

destinations will be computed to fill the destinations table. This tabulation of information will be convenient to generate the hypotheses by linking the corresponding cells of sources and destinations in the tables programmatically.

ii. Deficient Suspected Components:

It the set of *Deficient Suspected Components* that have been wrongly expected in a discrepancy state. In the previous demonstration example, the deficient suspected component of x_{d1} is $x_{d1,\{WRNG\}} = x_{d1,\{3\}} = \{V_3\}$. Thus, after evaluating the candidate states for each discrepancy state estimate x_{dj} in Z_d, the deficient suspected components will be computed by comparing each candidate state $x_{cndd\_k}(x_{dj})$ with its corresponding discrepancy state x_{dj} to determine the wrong part, where "k" is used to index the candidate states for each x_{dj}.

In the previous example comparing $x_{cndd\_1}(x_{d1})$= (O,Po,SO,Po,Po) compared to x_{d1}=(O,Po,**O**,Po,Po), then the deficient suspected component is determined by the wrong part which is V3. By this way, it will be possible to fill in a table that is called the deficient components table as follows:

Table 4.10: Deficient Components Table Demonstration Example

x_{dj}	$x_{cndd\_k}(x_{dj})$	$DSC_{j,k}$
x_{d1}=(O,Po,O,Po,Po)	(O,Po,SO,Po,Po)	{V3}
	(O,Po,O,Po,SO)	{V5}
x_{d2}=(O,O,Po,Po,O)	(O,SO,Po,Po,O)	{V2}
	(O,O,Po,Po,SO)	{V5}

- *The formulation:*

For each state estimate x_{dj} in the Discrepancy state Z_d, and for each candidate state, the deficient suspected components could be represented in the form of the following function:

$$\boldsymbol{DSC}(x_{cndd\_k}(x_{dj})) = \{c \mid c \in \{WRNG\}\}, \forall\ 1 \le j \le |Z_d|, \forall\ 1 \le k \le |X_{cndd}(x_{dj})|$$

And could be abbreviated as $DSC_{j,k}$

For some extreme cases if $|CRCT|= 0$, this cardinality of 0 means that we could not find a candidate state with a correct part, then $X_{cndd}(x_{dj}) = \{x_{ss} \in SS: |\{WRNG\}|=|C|\}$, where the size of the wrong part equals the number of components. Thus, **IF APPLICABLE**, we will select the fully abnormal states like for example (SO,SO,SO,SO,SO) as candidates, where all the components are wrongly expected.

b. Determine the Destinations of Missing Transitions

i. Destinations Table:

This table is computed based on Table 4.10 of deficient suspected components. Each field in the destinations table contains the candidate destination state of the component C_i per each x_{dj} (Based on **UnObRule-6**). The destination of a transition is formulated in the form a colon pair as $C_i{:}x_t$ where C_i is the component name and x_t is the state value of that component at its destination. The formula to get each field in this destinations table is as follows.

- *The formulation*:

For a component C_i indexed with "i" in a candidate state $x_{cndd\_k}(x_{dj})$ for a state estimate x_{dj} in the discrepancy state z_d, the destinations could be represented by the following formula:

$$Dest_{j,k,i} = Ci{:}\ x_{cndd\_k,i}(x_{dj}) \text{ if } Ci \in DSC_{j,k}\ \forall\ 1 \le j \le |z_d|,\ \forall\ 1 \le k \le |X_{cndd}(x_{dj})|,\ \forall\ 1 \le i \le |C|$$

This means for each state estimate x_{dj} in DS, there will be some candidate states $Xcndd(x_{dj})=\{x_{cndd\_1}(x_{dj}), \ldots, x_{cndd\_k}(x_{dj})\}$. In these candidate states, the value of the component C_i is the destination of the transition if this C_i belongs to the deficient components $DSC_{j,k}$ of this candidate state $x_{cndd\_k}(x_{dj})$.

For the previous example the destinations table will be constructed as follows:

Table 4.11: Destinations Table Demonstration Example

x_{dj}	$x_{cndd\_k}(x_{dj})$	$DSC_{j,k}$	V1	V2	V3	V4	V5
x_{d1}=(O,Po,O,Po,Po)	$x_{cndd\_1}(x_{d1})$=(O,Po,SO,Po,Po)	{V3}	---	---	V3:SO	---	---
	$x_{cndd\_2}(x_{d1})$=(O,Po,O,Po,SO)	{V5}	---	---	---	---	V5:SO
x_{d2}=(O,O,Po,Po,O)	$x_{cndd\_1}(x_{d2})$=(O,SO,Po,Po,O)	{V2}	---	V2:SO	---	---	---
	$x_{cndd\_1}(x_{d2})$=(O,O,Po,Po,SO)	{V5}	---	---	---	---	V5:SO

c. Determine the Sources of Missing Transitions

ii. Sources Table:

The source of missing transitions in the suspected components will be written in the form of colon pairs as $Ci{:}x_{dj,i}$ (based on **UnObRule-5**). The first element in the colon pair is the Deficient Suspected Component and the second element is state value of this component in the discrepancy state estimate x_{dj}. To backwardly link the destinations table to the sources, the source of missing transition will be computed for each deficient component cell in the Destinations Table $Dest_{j,k,i}$ using the following formula.

- *The formulation:*

For a component C_i indexed with "i" in a candidate state $x_{cndd\_k}(x_{dj})$ for a state estimate x_{dj} in the discrepancy state z_d, the sources could be given by the following function:

$Srcs(Dest_{j,k,i}) = C_i{:}x_{dj,i}$ $IF\ C_i \in DSC_{j,k}$ $\forall\ 1 \leq j \leq |z_d|, \forall\ 1 \leq k \leq |Xcndd(x_{dj})|, \forall\ 1 \leq i \leq |C|$

For the previous example the sources table will be constructed as follows:

Table 4.12: Sources Table Demonstration Example

x_{dj}	$x_{cndd\_k}(x_{dj})$	$DSC_{j,k}$	V1	V2	V3	V4	V5
x_{d1}=(O,Po,O,Po,Po)	$x_{cndd\_1}(x_{d1})$=(O,Po,SO,Po,Po)	{V3}	---	---	V3:Po	---	---
	$x_{cndd\_2}(x_{d1})$=(O,Po,O,Po,SO)	{V5}	---	---	---	---	V5:Po
x_{d2}=(O,O,Po,Po,O)	$x_{cndd\_1}(x_{d2})$(O,SO,Po,Po,O)	{V2}	---	V2:O	---	---	---
	$x_{cndd\_1}(x_{d2})$=(O,O,Po,Po,SO)	{V5}	---	---	---	---	V5:O

3- STEP-3: Hypotheses generation and model compliance verification

a. Generating the Hypotheses

i. Hypotheses

After computing the sources and destinations tables, it is possible to interlink between the corresponding cells in the tables to generate the hypotheses (**UnObRule-7 to 10**). As mentioned before such tabulation is beneficial in organizing the work to get it programmatically implemented through programming loops.

The hypotheses are computed for each deficient component C_i per each candidate state of each discrepancy state x_{dj}. Each hypothesis starts for the source of the missing transition to the candidate destination. The unobservable event is chosen according to the ***applicable*** unobservable event that leads to reach the candidate abnormal state. For example, if the destination candidate for a transition ended with a stuck-closed fault state in valve V1, the unobservable event is taken as V1SC. In the following formula, the unobservable event will be denoted as σ_{uo}.

ii. The simple hypothesis

For a component is starting from the source to the destination by the event of discrepancy in the component itself.

- *The formulation*:

$$H_{j,k,i} = Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_{uo}} Dest_{j,k,i}\ \forall\ 1 \leq j \leq |z_d{=}DS|, \forall\ 1 \leq k \leq |Xcndd(x_{dj})|, \forall\ 1 \leq i \leq |C|$$

For the previous demonstration example: if we assumed that in the valves, the fault state SO is reachable by the unobservable event VnSO where n is the valve number, then:

$$H1,1,3 = \text{V3:Po} \xrightarrow{\text{V3SO}} \text{V3:SO}$$

$$H1,2,5 = \text{V5:Po} \xrightarrow{\text{V5SO}} \text{V5:SO}$$

$$H2,1,2 = \text{V2:O} \xrightarrow{\text{V2SO}} \text{V2:SO}$$

$H2,2,5 = V5{:}O \xrightarrow{V5SO} V5{:}SO$

iii. Combined Hypothesis:

A hypothesis might be comprised of more than one transition at a time if the event of discrepancy is common and was missing in more than one component. Thus, for each state estimate, the hypothesis is generated as the combination of the simple transition hypotheses ending by the corresponding *candidate expected states*.

- *The formulation*:

$H_{j,k}$ = AND $[Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_{uo}} Dest_{j,k,i}] \; \forall \; 1 \leq j \leq |\mathcal{Z}_d = DS|, \; \forall \; 1 \leq k \leq |Xcndd(x_{dj})|, \; \forall \; 1 \leq i \leq |C|$

b. Synchronizing the corrected component and generating new models.

The previous formulas may result in a set of hypotheses for the corrected components. Consequently, the corrected components shall undergo a new synchronous product that generates a new set of hypothesized models. This set of models have their progression monitored by the observer. If any of new models becomes incompliant with the true plant observations while there are other models complying with the plant data, the incompliant model shall be rejected. This rejection is based on the fact that, there are some models having explanation for the plant data while the rejected model perhaps was built using incorrect hypothesis and it is the time to reject it.

- *The formulation*:

If a number of hypotheses M indexed by m are generated and the number of components is $|C|$ thus, the hypothesized models are given by:

$Model_m$= Synch $(C_1, C_2, \ldots, C_c)$, $1 \leq m \leq M$.

4.5.4.2 Unobservable Event Algorithm Pseudo Code

Algorithm 2: Unobservable Events Learning Algorithm

Input: Current model, $\boldsymbol{\sigma_d}$,**ALD**, **LCS**, **DS**.

Output: Set of hypothesized models.

1: Suspected States $\boldsymbol{SS} = \{x \in X_F \mid \lambda(x) = y_d\}$

2: **For each** $x_{dj} \in \mathcal{Z}_d$

3: Candidate States $\boldsymbol{Xcndd(x_{dj})} = \{x_{ss} \in \text{SS} \mid x_{ss,\{CRCT\}} = x_{d,\{CRCT\}}\}$, $\forall \, x_{dj} \in \mathcal{Z}_d$

4: AND $|\{CRCT\}|$ is maximum

5: AND $x_{ss,\{WRNG\}}$ belongs to faulty states.

6: AND Fault states in components are permanent.

7: **End**

8: /* Deficient Components

9: **For each** $x_{dj} \in z_d$

10: **For each** $x_{cndd\_k}(x_{dj})$

11: $DSC_{j,k} = \boldsymbol{DSC}(x_{cndd\_k}(x_{dj})) = \{c \mid c \in \{WRNG\}$

12: Fill deficient components table with $DSC_{j,k}$

13: **End**

14: **End**

15: /* Destinations Table

16: **For each** $x_{dj} \in z_d$

17: **For each** $x_{cndd\_k}(x_{dj})$ in $X_{cndd\_k}(x_{dj})$

18: **For each** C_i in $x_{cndd\_k}(x_{dj})$

19: $Dest_{j,k,i} = Ci{:}\ x_{cndd\_k,i}(x_{dj})$ if $Ci \in DSC_{j,k}$

20: Fill Destinations Table with $Dest_{j,k,i}$

21: **End**

22: **End**

23: **End**

24: /* Sources Table

25: **For each** $x_{dj} \in z_d$

26: **For each** $x_{cndd\_k}(x_{dj})$ in $X_{cndd\_k}(x_{dj})$

27: **For each** C_i in $x_{cndd\_k}(x_{dj})$

28: $Srcs(Dest_{j,k,i}) = C_i{:}x_{dj,i}$ *IF* $C_i \in DSC_{j,k}$

29: Fill in Sources Table with $Srcs(Dest_{j,k,i})$

30: **End**

31: **End**

32: **End**

33: /* Generate the Hypotheses

34: **For each** $x_{dj} \in z_d$

35: **For each** $x_{cndd\_k}(x_{dj})$ in $X_{cndd\_k}(x_{dj})$

36: **For each** C_i in $x_{cndd\_k}(x_{dj})$

37: $H_{j,k,i} = Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_{uo}} Dest_{j,k,i}$

38: **End**

39: **End**

40: **End**

41: /* Combined Hypotheses

42: $H_k(x_{dj})$ = AND $[Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_{uo}} Dest_{j,k,i}]$ $\forall$ $1 \leq j \leq |z_d = DS|$, $\forall$ $1 \leq k \leq |Xcndd(x_{dj})|$, $\forall$ $1 \leq i \leq |C|$

43: /* Synchronize the corrected components and generate the new models

44: **For each** H in $H_{j,k,i}$

45: $Model_m$= Synch $(C_n)_m$, $1 \leq n \leq |C|$, $1 \leq m \leq M$, M is the number of hypotheses.

46: **End**

4.5.4.3 Application Example-2

The following example describes the application of the algorithm after detecting a discrepancy up to generating the hypotheses for the missing transitions with **unobservable** events in the model. In this example, the plant is comprised of three similar cascaded valves. The automaton of the first valve, the control sequence and the plant are shown in Figure 4.32. The model of the plant was built by the synchronous product of the components' automata. We deleted the transition with V1SC, V2SC and V3SC between O and SC in V1, V2 and V3 as indicated by the crossed transition in Figure 4.32. This example will tackle the case of type-2 of discrepancy where the system generated a new unexpected output label with no observable events were applied.

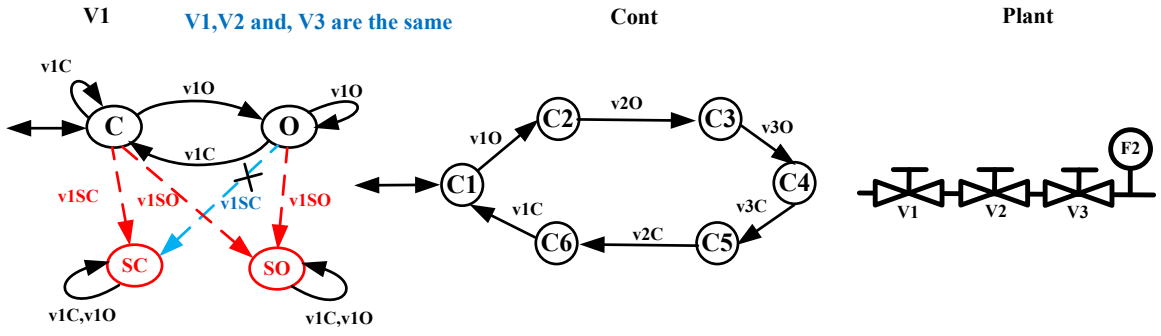


Figure 4.32: Application Example-2 on Unobservable Events

z_0= {1,3,4,5,6,7,8}

y_0 = fL

➔ **V1O , y_1 = fL:** new event occurred at the real system with no label change (Case-1)

Since $z_{k+1} = \xi_\sigma (z_k, \sigma_{k+1})$, $\xi_\sigma (z_k, \sigma_{k+1}) = \{x \mid x \in UR (\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$

Then, $z_1 = \{x \mid x \in UR (\eta(z_0, \mathbf{V1O}))$ and $\lambda(x)=\mathbf{fL}\}$.

$x \in z_0$	$\eta(x,\mathbf{V1O})$	*UR*(.)	*UR*(.) and λ(*x*)=fL
1	$\eta(1,\mathbf{V1O})=2$	{2,9,11,12,13,14}	{2,9,11,12,13,14}
3	$\eta(3,\mathbf{V1O})=15$	{15,43,44,45,46}	{15,43,44,45,46}
4	$\eta(4,\mathbf{V1O})=9$	{9,29,30,31,32}	{9,29,30,31,32}
5	$\eta(5,\mathbf{V1O})=11$	{11,29,38,39}	{11,29,38,39}
6	$\eta(6,\mathbf{V1O})=12$	{12,30,40,41}	{{12,30,40}
7	$\eta(7,\mathbf{V1O})=13$	{13,31,38,40}	{13,31,38,40}
8	$\eta(8,\mathbf{V1O})=14$	{14,32,39,41}	{14,32,39}

Thus, $z_1 = \{2,9,11,12,13,14,15,29,30,31,32,38,39,40,43,44,45,46\}$

➔ **V2O , y_2 = fL:** new event occurred with no label change (Case-1)

Then, $z_1 = \{x \mid x \in UR (\eta(z_0, \mathbf{V2O}))$ and $\lambda(x)=\mathbf{fL}\}$.

$x \in z_1$	$\eta(x,\mathbf{V2O})$	*UR*(.)	*UR*(.) and λ(*x*)=fL
	$\eta(2,\mathbf{V2O})=10$	{10,28,33,35,36}	{10,28,33,35}
	$\eta(9,\mathbf{V2O})=28$	{28,55,57,58}	{28,55,57}
	$\eta(11,\mathbf{V2O})=37$	{37,59,71,72}	{37,59,71,72}
	$\eta(12,\mathbf{V2O})=33$	{33,55,65,66}	{33,55,65}
	$\eta(13,\mathbf{V2O})=35$	{35,57,65}	{35,57,65}
	$\eta(14,\mathbf{V2O})=36$	{36,58,66}	---
	$\eta(15,\mathbf{V2O})=42$	{42,73,75,76}	{42,73,75,76}
	$\eta(29,\mathbf{V2O})=59$	{59,89,90}	{59,89,90}
	$\eta(30,\mathbf{V2O})=55$	{55,83,84}	{55,83}
	$\eta(31,\mathbf{V2O})=57$	{57,83}	{57,83}
	$\eta(32,\mathbf{V2O})=58$	{58,84}	---
	$\eta(38,\mathbf{V2O})=71$	{71,89}	{71,89}
	$\eta(39,\mathbf{V2O})=72$	{72,90}	{72,90}
	$\eta(40,\mathbf{V2O})=65$	{65,83}	{65,83}

	η(43,**V2O**)=77	{77,107,108}	{77,107,108}
	η(44,**V2O**)=73	{73,101,102}	{73,101,102}
	η(45,**V2O**)=75	{75,101}	{75,101}
	η(46,**V2O**)=76	{76,102}	{76,102}

Thus, z_2={10,28,33,35,37,42,55,57,59,65,71,72,73,75,76,77,83,89,90,101,102,107,108}

➔ **V3O , y_3 = fH:** new event occurred with label change (Case-3)

Since $z_{k+1} = \xi_e(z_k, \sigma_{k+1}, y_{k+1})$, $\xi_e(z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_{k+1}\}$.

Then, $z_3 = \{x \mid x \in UR(\eta(z_2,$ **V3O**$))$ and $\lambda(x)=$**fH**$\}$

$x \in z_2$	$\eta(x \in$,**V3O**)	***UR*(.)**	***UR*(.) and $\lambda(x)$=fH**
10	η(10,**V3O**)=34	{34,56,64,68}	{34,56,64,68}
28	η(28,**V3O**)=56	{56,82,86}	{56,82,86}
33	η(33,**V3O**)=64	{64,82,92}	{64,82,92}
35	η(35,**V3O**)=69	{69,87,93}	---
37	η(37,**V3O**)=70	{70,88,98}	---
42	η(42,**V3O**)=74	{74,100,104}	---
55	η(55,**V3O**)=82	{82,110}	{82,110}
57	η(57,**V3O**)=87	{87,111}	---
59	η(59,**V3O**)=88	{88,116}	---
65	η(65,**V3O**)=93	{93,111}	---
71	η(71,**V3O**)=99	{99,117}	---
72	η(72,**V3O**)=98	{98,116}	---
73	η(73,**V3O**)=100	{100,127}	---
75	η(75,**V3O**)=105	{105,128}	---
76	η(76,**V3O**)=104	{104,127}	---
77	η(77,**V3O**)=106	{106,133}	---
83	η(83,**V3O**)=111	{111}	---
89	η(83,**V3O**)=117	{117}	---
90	η(90,**V3O**)=116	{116}	---
101	η(101,**V3O**)=128	{128}	---

102	η(102,**V3O**)=127	{127}	---
107	η(107,**V3O**)=134	{134}	---
108	η(108,**V3O**)=133	{133}	---

Thus, z_3 = {34,56,64,68,82,86,92,110}.

➔ V2SC unobservable event occurred in the true system.

y_4 = **fL:** new label without an observable event (Case-2)

$z_{k+1} = \xi_y(z_k, y_{k+1})$, $\xi_y(z_k, y_{k+1}) = \{x \mid \lambda(x) = y_{k+1} \text{ and } x \in UR(z_k)\}$.

Then, $z_4 = \{x \mid \lambda(x) = \textbf{fL} \text{ and } x \in UR(z_3)\}$.

Since z_3 = {34,56,64,68,82,86,92,110}

$x \in z_3$	***UR*(.)**	***UR*(.) and λ(*x*) = fL**
34	{34,56,64,68}	---
56	{56,82,68}	---
64	{64,82,92}	---
68	{68,86,92}	---
82	{82 ,110}	---
86	{86,110}	---
92	{92,110}	---
110	{110}	---

This means that, for this label change, if we started from the states of z_3 = {34, 56, 64, 68, 82,86,92,110} we cannot reach any state having label = fL by an unobservable event. Thus, the new label cannot be explained by the current model, and a DISCREPANCY IS DETECTED.

STEP1: Detect the discrepancy and determine the discrepancy type.

a. **Detect Discrepancy and its Type**

DISCREPANCY IS DETECTED.

i. **Type of Discrepancy**: Unexpected output label

ii. **Cause:** Missing transitions with unobservable event

b. **Collect the Current System Observations and Information**

i. Actual Label of Discrepancy **ALD**= y_d= **fL**

ii. Last Conforming State **LCS:** z_L= z_3 = {34,56,64,68,82,86,92,110}

Here, in this type of discrepancy, the discrepancy sate DS=LCS,

DS= z_3 = {34,56,64,68,82,86,92,110}

STEP-2: Analyzing the Collected Information and Concluding Sources and Candidates

a. Determine the Deficient Suspected Components:

i. Suspected States **SS:** from the output map table, we have the following states that can have a label of **fL**. For simplicity, the SS table will be represented in a compact representation such that if any valve is closed or stuck closed, the other valves could be in any state to get a label of **fL**. The sign "/" in the table means "or", and this is to compress the number of rows to make the table readable.

Table 4.13: Application Example-2 SS Table

V1	V2	V3	y
C	C/O/SC/SO	C/O/SC/SO	fL
SC	C/O/SC/SO	C/O/SC/SO	fL
C/O/SC/SO	C	C/O/SC/SO	fL
C/O/SC/SO	SC	C/O/SC/SO	fL
C/O/SC/SO	C/O/SC/SO	C	fL
C/O/SC/SO	C/O/SC/SO	SC	fL

ii. Candidate States ***Xcndd***

$\boldsymbol{Xcndd}(x_{dj}) = \{ x_{ss} \in SS \mid x_{ss,\{CRCT\}} = x_{dj,\{CRCT\}} \}$, $\forall\, x_{dj} \in Z_d$

From the SS-table, the candidate states for each x_{dj} are listed and the actual values of the x_{dj} are given.

Table 4.14: Application Example-2 Candidate States

$DS = Z_3$	$\boldsymbol{Xcndd}(x_{dj})$
34 = (O,O,O)	(SC,O,O), (O,SC,O),(O,O,SC)
56 = (SO,O,O)	(SO,SC,O), (SO,O,SC)
64 = (O,SO,O)	(SC,SO,O),(O,SO,SC)
68 = (O,O,SO)	(SC,O,SO),(O,SC,SO)
82 = (SO,SO,O)	(SO,SO,SC)
86 = (SO,O,SO)	(SO,SC,SO)
92 = (O,SO,SO)	(SC,SO,SO)
110 = (SO,SO,SO)	---

As shown in Table 4.14, the candidates are selected according to the conditions of selecting the candidates described in the algorithm formulation. For example, for the state 68 = (O,O,SO), the $|\{CRCT\}|$ is selected to be maximum, so the selected candidates have only one component changed at a time. Also, the component changed to a faulty state, while the faulty state SO remained unchanged.

iii. Deficient Suspected Components:

Table 4.15: Application Example-2 Deficient Components

x_{dj}	$x_{cndd\_k}(x_{dj})$	$DSC_{j,k}$
34 = (O,O,O)	(SC,O,O)	{V1}
	(O,SC,O)	{V2}
	(O,O,SC)	{V3}
56 = (SO,O,O)	(SO,SC,O)	{V2}
	(SO,O,SC)	{V3}
64 = (O,SO,O)	(SC,SO,O)	{V1}
	(O,SO,SC)	{V3}
68 = (O,O,SO)	(SC,O,SO)	{V1}
	(O,SC,SO)	{V2}
82 = (SO,SO,O)	(SO,SO,SC)	{V3}
86 = (SO,O,SO)	(SO,SC,SO)	{V2}
92 = (O,SO,SO)	(SC,SO,SO)	{V1}

b. Determine the Destinations of Missing Transitions

i. Destinations Table:

$$Dest_{j,k,i} = Ci: x_{cndd\_k,i}(x_{dj}) \text{ if } Ci \in DSC_{j,k} \forall\ 1 \le j \le |\mathcal{Z}_d = DS|, \forall\ 1 \le k \le |Xcndd(x_{dj})|, \forall\ 1 \le i \le |C|$$

Table 4.16: Application Example-2 Destinations Table

x_{dj}	$x_{cndd\_k}(x_{dj})$	$DSC_{j,k}$	V1	V2	V3
34 = (O,O,O)	(SC,O,O)	{V1}	V1:SC		
	(O,SC,O)	{V2}		V2:SC	
	(O,O,SC)	{V3}			V3:SC
56 = (SO,O,O)	(SO,SC,O)	{V2}		V2:SC	
	(SO,O,SC)	{V3}			V3:SC
64 = (O,SO,O)	(SC,SO,O)	{V1}	V1:SC		
	(O,SO,SC)	{V3}			V3:SC
68 = (O,O,SO)	(SC,O,SO)	{V1}	V1:SC		
	(O,SC,SO)	{V2}		V2:SC	
82 = (SO,SO,O)	(SO,SO,SC)	{V3}			V3:SC
86 = (SO,O,SO)	(SO,SC,SO)	{V2}		V2:SC	
92 = (O,SO,SO)	(SC,SO,SO)	{V1}	V1:SC		

c. Determine the sources of missing transitions

i. Sources Table:

$$Srcs(Dest_{j,k,i}) = C_i: x_{dj,i} \text{ if } C_i \in DSC_{j,k} \ \forall\ 1 \le j \le |\mathcal{Z}_d = DS|, \forall\ 1 \le k \le |Xcndd(x_{dj})|, \forall\ 1 \le i \le |C|$$

The sources table will be constructed as follows:

Table 4.17: Application Example-2 Sources Table

x_{dj}	$x_{cndd\_k}(x_{dj})$	$DSC_{j,k}$	V1	V2	V3
34 = (O,O,O)	(SC,O,O)	{V1}	V1:O		
	(O,SC,O)	{V2}		V2:O	
	(O,O,SC)	{V3}			V3:O
56 = (SO,O,O)	(SO,SC,O)	{V2}		V2:O	
	(SO,O,SC)	{V3}			V3:O
64 = (O,SO,O)	(SC,SO,O)	{V1}	V1:O		
	(O,SO,SC)	{V3}			V3:O
68 = (O,O,SO)	(SC,O,SO)	{V1}	V1:O		
	(O,SC,SO)	{V2}		V2:O	
82 = (SO,SO,O)	(SO,SO,SC)	{V3}			V3:O
86 = (SO,O,SO)	(SO,SC,SO)	{V2}		V2:O	
92 = (O,SO,SO)	(SC,SO,SO)	{V1}	V1:O		

STEP-3: Hypotheses generation and model compliance verification

a. Generating the Hypotheses

Hypotheses:

$H_{j,k,i} = Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_{uo}} Dest_{j,k,i}\ \forall\ 1 \le j \le |z_d = DS|,\ \forall\ 1 \le k \le |Xcndd(x_{dj})|,\ \forall\ 1 \le i \le |C|$

The generated hypotheses are:

$H1,1,1 =$ V1:O $\xrightarrow{\text{V1SC}}$ V1:SC

$H1,1,2 =$ V2:O $\xrightarrow{\text{V2SC}}$ V2:SC

$H1,1,3 =$ V3:O $\xrightarrow{\text{V3SC}}$ V3:SC

$H2,1,2 =$ V2:O $\xrightarrow{\text{V2SC}}$ V2:SC Repeated

$H2,2,3 =$ V3:O $\xrightarrow{\text{V3SC}}$ V3:SC Repeated

$H3,1,1 =$ V1:O $\xrightarrow{\text{V1SC}}$ V1:SC Repeated

$H3,1,3 =$ V3:O $\xrightarrow{\text{V3SC}}$ V3:SC Repeated

$H4,1,1 =$ V1:O $\xrightarrow{\text{V1SC}}$ V1:SC Repeated

$H4,1,2 =$ V2:O $\xrightarrow{\text{V2SC}}$ V2:SC Repeated

$H5,1,3 =$ V3:O $\xrightarrow{\text{V3SC}}$ V3:SC Repeated

$H6,1,2 =$ V2:O $\xrightarrow{\text{V2SC}}$ V2:SC Repeated

$H7,1,1 =$ V1:O $\xrightarrow{\text{V1SC}}$ V1:SC Repeated

The repeated hypotheses having the same transition to correct a component will be ignored this result that we have three unique hypotheses.

$H1,1,1 =$ V1:O $\xrightarrow{\text{V1SC}}$ V1:SC

$H1,1,2 =$ V2:O $\xrightarrow{\text{V2SC}}$ V2:SC

$H1,1,3 =$ V3:O $\xrightarrow{\text{V3SC}}$ V3:SC

In this example, the three hypotheses are equally possible in their effect such that any one of them if has been added to the plant can explain the discrepancy. This means that they will lead to generating three compliant models where any of them could be selected or it could be left up to the designer's decision to combine the hypotheses.

4.6 Diagnosis with Multiple Hypotheses

In normal operation, the observer evaluates the states estimates of the system and forwards these estimated states to the diagnoser's condition map to compute the system condition either normal or in fault mode. As discussed previously, the event-state observer detects the discrepancy, and the learning algorithm works on generating hypotheses for the missing transitions to amend the model and to update the observer.

After the generation of a set of hypotheses, there will be a number of models that correspond to the generated hypotheses. In this case, the final condition of the system will be taken as the union of the conditions computed of all the current hypothesis as shown in Figure 4.33. Based on the compliance of the hypotheses, the incompliant hypotheses are rejected, while the compliant ones are kept. The union of conditions is taken as the ensemble of the decisions of each single hypothesis. This is in fact, similar to the ensemble technique used in machine learning when it is required to combine the decision of multiple learners [42].

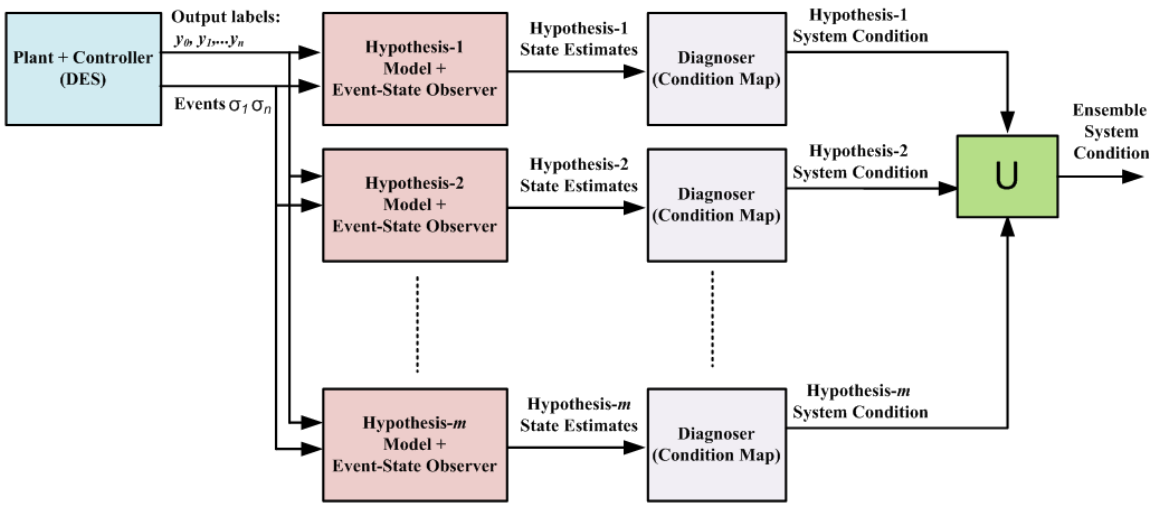


Figure 4.33: Diagnosis with Multiple Hypotheses

4.7 Computational Complexity

4.7.1 Introduction

Since a certain problem could be theoretically solvable but in practice it may not be possible to solve it due to the higher requirements of time and computer resources. Computational complexity is introduced as an important measure to analyze algorithms in terms of time and memory [53]. The time complexity is independent of the hardware whether the algorithm is executed on a nano computer or supercomputer [54]. In this thesis we will study only the time complexity for the learning diagnoser algorithm.

As the exact computational time of an algorithm could be a very complex task, the study of computational complexity of an algorithm aims to estimate it under its worst case. Such an estimation is called the asymptotic analysis that seeks an understanding of the runtime while the inputs grow to be very large [53]. The "Big-*O*" notation is used to represent the time complexity of algorithms while running on large inputs and could define the upper bound of the complexity of the algorithm. For example, considering an algorithm with its complexity was determined to be $O(f(N))$, then it is said that $f(N)$ is an upper bound of this algorithm complexity.

Since each basic single computation statement such as addition, subtraction will be considered to consume a unit of time, the "Big-*O*" is considered as an estimation of the number of time unites consumed in the algorithm. In computing the "Big-*O*", the lower order terms and coefficients are ignored, and the

highest order term of the expression will be considered. This is because the highest term will dominate the other lower terms and then the "Big-O" describes the complexity of the highest term [53].

4.7.2 Techniques

The following techniques are considered to be applied in the estimation process of computational complexity.

1- The very small loops such as of two iterations to check flags will be ignored and considered as fixed sized constant statements and is not affected by input size growth.
2- The cardinality of any subsets used in the algorithm is replaced by or approximated to the cardinality of its superset that could be concluded from the inputs and this is to consider the worst case. For example, the cardinality of the proposed states set X_{PS} is replaced by the cardinality of the model states set X, since the bigger the model the bigger the proposed states. This is because the cardinality of these supersets can be concluded while running the algorithm, and the worst case is considered if the set equals its super set.
3- Since the model is comprised of a set of components, and for the sake of estimation, all the components are assumed to be the same, i.e., all components' models are assumed to have the same automaton. This implies that during this estimation, the number of states, transition list length and the number events are the same for all the components.

4.7.3 Learning Algorithm Computational Complexity

In this section, the computational complexity of the learning algorithm is discussed. As was shown and discussed earlier in the general learning algorithm flow chart, based on the discrepancy type, there are two sub-algorithms. The first treats the missing observable events represented by Algorithm-1 meanwhile the second sub-algorithm delas with the missing non observable events represented by Algorithm-2. Both of them have similar steps of collecting the current data, collected data analysis in terms of finding suspected components determining sources and destination of hypotheses, and finally to generate the new hypothesized models.

Since both algorithms exhibit almost the same behavior in computation, the first algorithm "Algorithm-1" is used to study the computational complexity.

- Determining the suspected components is estimated to have the following complexity.

$$O\left(|C|.|\Sigma_c|\right)$$

where $|C|$, is the number of components in the model and $|\Sigma_c|$ is the number of events in the event set of the component automaton.

- Determining complete and deficient suspected components is estimated to have the following complexity.

$$O\left(|\mathcal{Z}_L|.|T_L|.|SC|\right)$$

Where $|\mathcal{Z}_L|$ is the number of states estimates in a diagnoser evolving step, and $|T_L|$ is the number of transitions in the component automaton, and $|SC|$ is the number of suspected components.

This will be approximated to be

$$O\left(|X|.|T_L|.|C|\right)$$

Where $|X|$ is the number of states in the model.

- Determining candidate suspected states is estimated to have the following complexity.

$$O\left(|\mathcal{Z}_L|.|X_{PS}|.[|C_{irv}|+|T_L|.|CSC|+|C|+|X_c|]\right)$$

Where $|\mathcal{Z}_L|$ is the number of states estimates in a diagnoser evolving step, $|X_{PS}|$ is the number of states in the output map having output label = ALD, $|C_{irv}|$ is the number of irrelevant components, $|T_L|$ is the number of transitions in an automaton , $|CSC|$, is the number of complete suspected components, $|C|$ is the number of components, and $|X_c|$ is the number of states in automaton.

This will be approximated to be

$$O\left(|X|.|X|.[|C|+|T_L|.|C|+|C|+|X_c|]\right)$$

$$\simeq O\left(|X|^2.[|T_L|.|C|+|X_c|]\right)$$

- Destination and Sources tables will be estimated to have

$$O\left(|X|^2.|C|^2\right)$$

- Sync Operation is used to generate the hypothesized models. In fact, this operation is found to be the most time consuming operation in the algorithm and it dominates the other steps of the algorithm.

This operation starts with computing the inverse projection of the components' automata, then it will operate the product operation on them. The recursive loop of operating the product of invers projected automata will finally be estimated to have

$$O\left(X_c^{|c|}\right)$$

where $|X_c|$ is the number of states of an automaton, and $|C|$ is the number of components in the model.

4.7.4 Diagnoser Update

As explained in the framework block diagram, once a discrepancy is detected, the learning algorithm will generate hypotheses and create hypothetical models. For a certain hypothetical newly generated model, the event-state observer should be updated to be able to deduce the state estimation based on the new model. In [20], the event-state observer is implemented based on the ***Reachability Transition System*** (RTS). For each state of a certain automaton, the reachability analysis complexity is $O\ (|X|+|T|)$ where X is the set of states, and T is the transition list. Thus, for the entire automaton the reachability could be obtained in $O\ (|X|^2+|X|.|T|)$. Therefore, it could be said that updating the diagnoser that corresponds to a certain generated model will have a complexity of

$$O\ (|X|^2+|X|.|T|)$$

Where $|X|$ and $|T|$ are the number of states and the transition list length of the entire model. $|T|$ could be evaluated as $|X|.|\Sigma|$, where $|\Sigma|$ is the cardinality of the events set of the entire model.

In terms of the components' notation, and since $X \simeq X_c^{|c|}$ as the result of the cartesian product of the states of all components, then updating the diagnoser will have complexity of

$$O\ (|X_c|^{2|c|}+|X_c|^{|c|}.|T|)$$

$$\simeq O\ (|X_c|^{2|c|})$$

4.7.5 Entire Learning Diagnoser Algorithm Complexity Estimation

So finally, as discussed above all the comprising entities of the framework where the exponential term of the sync operation dominates all the other terms of the other steps of the algorithm, then the diagnoser update dominates the sync operation. Hence, we can say that the algorithm has complexity of

$$O\ (|X_c|^{2|c|})$$

Where $|X_c|$ is the number of states of an automaton, and $|C|$ is the number of components in the model.

Chapter 5

Case Study: Ozone Generation System

In this chapter, the Ozone Generation System will be introduced as a case for applying the algorithm where one of the common modeling mistakes will be detected and treated using the learning algorithm. This mistake appears when the designer assumes a wrong interaction among components which leads to a missing transition in the nominal model. While the real plant is generating the events and output labels, the learning algorithm will be used to detect the discrepancy and to correct the model.

5.1 Ozone Generation System Overview

Ozone is used in some water treatment plants to disinfect potable water. It was firstly used in 1893 at Oudshoorn, Netherlands [52]. Figure 5.1 shows the schematic diagram of an ozone generation system. Briefly, ozone is generated from oxygen subjected to high voltage applied between two electrodes. The vessel in which this interaction takes place is called the ozone generator. As shown in the figure, the process comprises mainly of the ozone generator vessel. Oxygen is supplied by another subsystem and its flow is controlled by the inlet and outlet valves V2 and V3 respectively. The flow is monitored by a flow meter “F2” while the ozone concentration is monitored by an ozone analyzer “AIT” (Analyzer Indicator Transmitter).

A stream of cooling water is used to keep the temperature within the required limits to assure the good production efficiency of ozone. The cooling water stream is controlled by valve V1, and the water flow is monitored by flowmeter F1.

A power supply unit (PSU) is used to generate the high voltage required to generate ozone. The power supply unit, in conjunction with the cooling water supply, governs the quality of the generated ozone represented in terms of the ozone concentration. Both of the power supply unit and the cooling water must be operational to generate ozone as measured by the ozone concentration analyzer AIT. On top of these equipments is the supervisor that receives the readings and sends the commands based on the supervisory specifications.

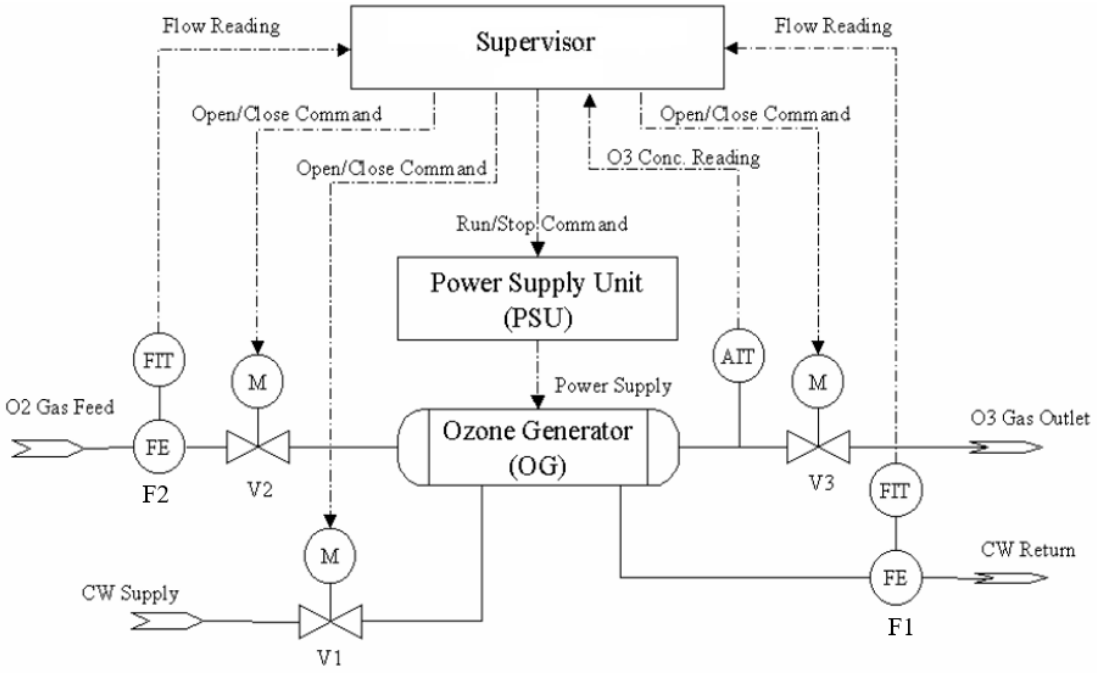


Figure 5.1: Ozone Generation System

5.2 Plant Model

5.2.1 Basic Components

Figure 5.2 shows the automata of the components in the system.

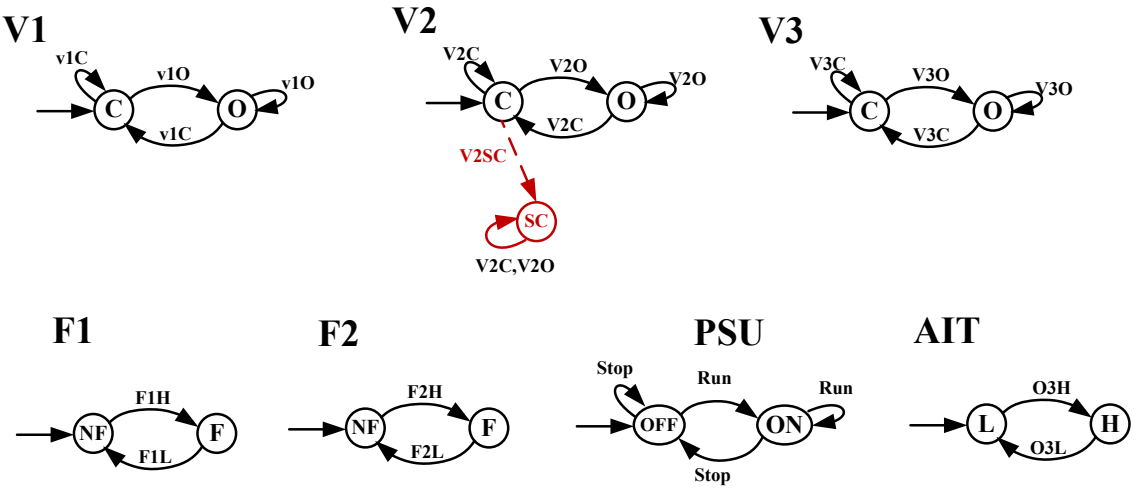


Figure 5.2: Ozone Generation System Components

The states and the events symbols for each automaton is described in the following.

Valve V1

V1 is the **cooling water valve** automaton, and it has two states, open (O) and closed (C). It has two events, open command (V1O) and close command (V1C).

Valve V2

V2 is the valve of the **oxygen inlet** automaton, and it has three states, open (O), closed (C), and stuck-closed (SC). It has three events, open command (V2O), close command (V2C), and stuck-closed failure (V2SC). As shown in the model, V2 can get stuck-closed from its closed state.

Valve V3

V3 is the valve of **ozone outlet** automaton, and it has two states, open (O) and closed (C). It has two events, open command (V3O) and close command (V3C).

Flowmeter F1

F1 is the **cooling water flowmeter** automaton, and it has two states, flow (F) and no-flow (NF). It has two events, high flow (F1H) and low flow (F1L).

Flowmeter F2

F2 is the **oxygen flowmeter** automaton, and it has two states, flow (F) and no-flow (NF). It has two events, high flow (F2H) and low flow (F2L).

Ozone Analyzer AIT

AIT is the **ozone analyzer indicator transmitter** automaton, and it has two states, low (L) and high (H). It has two events, high concentration (O3H) and low concentration (O3L).

Power Supply PSU

PSU is the **power supply unit** automaton, and it has two states, OFF and ON. It has two events, run command (Run), and stop command (Stop).

5.2.2 Interactions

Generally speaking, the interaction automata represent how some components are affected by the events happening in other components, e.g., the influence of operating valves on the flowmeter readings. Here, we have three interactions explained in the following.

Interaction-1: F1_V1:

The reading of F1 changes based on the state of V1. If V1 is open, the flow meter F1 will read high (H). This is indicated by a self-loop of F1H at the state "O" in V1. Similarly, if V1 is closed F1 will read low (L). This is indicated by a self-loop of F1L at the state "C" in V1. From another point of view, it could be said that the event F1H happens or permitted to happen when V1 is open and similarly, F1L happens when the valve is closed.

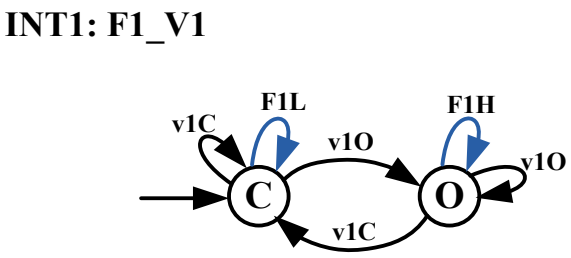


Figure 5.3: Interaction-1 F1_V1

Interaction-2 F2_V2_V3:

The reading of flow meter F2 is determined by the state of valves V2 and V3. The flowmeter will have a high reading if both valves are open otherwise the reading will be low. Since F2 depends on V2 and V3, we can model the interaction by first get the synchronous product of V2 and V3, then suitable F2 events will be added as self-loops to the resulting automaton: self-loop of F2H at state (O,O) where both valves are open; F2L self-loops at all other states (when at least one of the valves is closed). Figure 5.4 shows the results. The valve events are omitted from the figure to avoid cluttering the figure. The important information in this automaton is F2 self-loop transitions at the states.

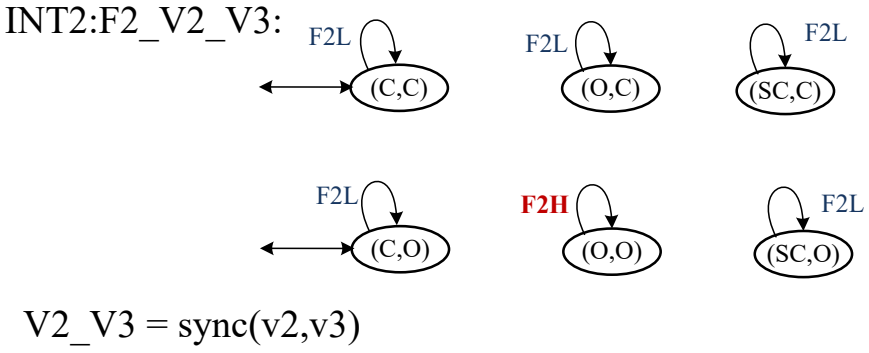


Figure 5.4: Interaction-2 F2_V2_V3

Interaction-3: AIT_F1_PSU_F2:

In reality the values of the ozone concentration analyzer depend on the states of the power supply unit (PSU) and the flow of cooling water F1. If the power supply is ON and the cooling water flow is high the analyzer can generate high reading. Otherwise, if any of these conditions is not met, the oxygen will not be properly transformed to ozone and the analyzer reading will be low. So, the correct modeling of this interaction is to get the synchronous product of the power supply automaton (PSU) with cooling water flowmeter automaton (F1), then to add self-loops of ozone analyzer (AIT) events accordingly.

A common mistake happens here - could be due to a wrong assumption – is when the designer assumes that the ozone concentration depends also on the flowmeter of oxygen. In this case, the designer obtains the synchronous product of F1,PSU and F2, then adds self-loops of ozone analyzer events.

In this case study, we assume the above modeling mistake is done and we see how it leads to a discrepancy during the operation of the plant and how the learning algorithm detects and generates a hypothesis to add the missing transition. Figure 5.5 below shows the automaton of the incorrect interaction AIT_F1_PSU_F2. This interaction is modeled as an automaton by forming the synchronous product of F1, PSU and F2, then adding self-loops of AIT events. It should be noted that events of F1, PSU, and F2 are omitted to avoid cluttering the figure (since our focus is on the events of the ozone analyzer). As shown in the figure, state (F,ON,F), corresponding to (F1: Flow, PSU:ON, F2:Flow), has a self-loop of O3H. All the other states have the self-loop of O3L.

State (F,ON,NF) in particular corresponding to (F1: Flow, PSU:ON, F2: No Flow), has an O3L self-loop, however in reality, having cooling water flow and PSU in ON state should permit the O3H to happen because even if there is no more oxygen supply but still the remaining oxygen in the vessel could be transformed to ozone. This missing self-loop transition of O3H at (F,ON,NF) is the missing transition in this example. Note that, there is an incorrect transition of O3L at this state, but the scope of our algorithm is to add any missing transitions and not to remove wrong transitions.

In general, in a deterministic setup as ours, discrepancies can be used to identify missing transitions, not to eliminate existing ones. To eliminate transitions, one can use a probabilistic framework or examine a more detailed model of components. In our case, once the learning algorithm adds an O3H self-loop to (F,ON,NF), the designer is alerted to the fact that perhaps O3L self-loop at the same state was an error.

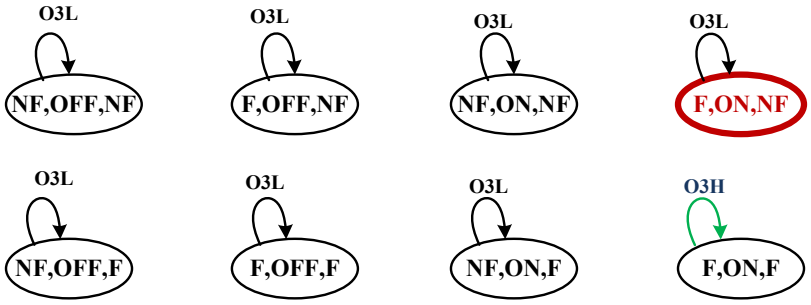


Figure 5.5: Interaction-3 AIT_F1_PSU_F2

The Plant model is generated by having the synchronous product of the components and the interactions as follows:

Plant = Sync (V1, V2, V3, PSU, F1, F2, AIT, INT1, INT2, INT3)

5.2.3 Supervisor

The supervisor block is modular, consists of two automata that enforce two specifications.

SUP1:

The first specification is the startup and shut down sequences:

Startup sequence:

- Open the cooling water stream valve (V1)
- Open the inlet oxygen valve (V2)
- Open the outlet oxygen valve (V3)
- Start the power supply unit (PSU)

Shutdown sequence:

- Stop the power supply unit (PSU)
- Close the cooling water stream valve (V1)
- Close the inlet oxygen valve (V2)
- Close the outlet oxygen valve (V3)

Both sequences are enforced by the supervisor module in Figure 5.6. All other events that are not restricted by this supervisory module are permitted and thus added as self-loops at all supervisor states. To avoid cluttering the figure, the self-loops are indicated under the figure.

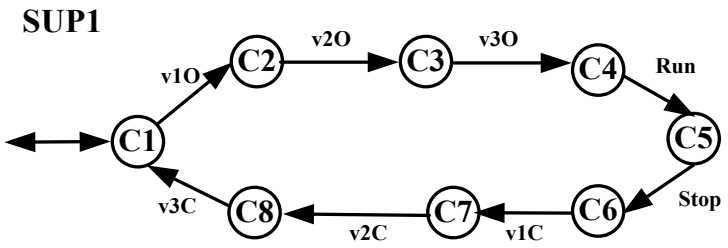


Figure 5.6: Startup and Shutdown Supervisory Specification SUP 1

SUP2:

The second safety specification requires that the ozone generator vessel be purged after the shutdown. Thus, both valves V2 and V3 will not be closed until the ozone concentration falls low. This could be modeled by the automaton in Figure 5.7. Here, a copy of ozone analyzer automaton is used to monitor the concentration and in its low state, both events of v2C and v3C are permitted (i.e., self-loops at state L). However, at H state, both v2C and v3C are absent (i.e., not permitted by the supervisor). Also, all other events that are not restricted by this supervisory specification are permitted (i.e., self-loops at all states. To avoid cluttering the figure, the self-loops are indicated under the figure.

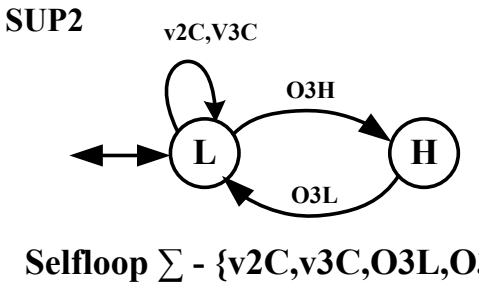


Figure 5.7: Supervisory Specification SUP 2

The supervisor denoted as SUP will be computed by the product of SUP1 and SUP2 as follows:

SUP = prod(Sup1,Sup2)

The entire Model of the system under supervision denoted as OGS, is computed by the product of the plant model and the supervisor.

OGS = prod(Plant, SUP)

The model under supervision has 70 states and 151 transitions.

5.3 System in Operation

Next we discuss a scenario which results in discrepancy and see how the learning algorithm works. The output labels will be represented in terms of the readings of F1, F2 and AIT respectively. Also, all events are observable except the stuck-closed failure of valve 2.

The system is initialized, and the observer starts with initial state z_0

Since $z_0 = \{x \mid x \in UR(\{x_0\}) \text{ and } \lambda(x) = \lambda(x_0)\}$, $x_0 = 1$, then

$z_0 = \{x \mid x \in UR(\{1\}) \text{ and } \lambda(x) = (\text{NF,NF,L})\}$

$z_0 = \{1,3\}$

$y_0 = (\text{NF,NF,L})$

➔ **V1O , y_1 = (NF,NF,L):** new event occurred at the true system with no label change (Case-1)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1})) \text{ and } \lambda(x) = y_k\}$

Then, $z_1 = \{x \mid x \in UR(\eta(z_0, \mathbf{V1O})) \text{ and } \lambda(x) = (NF,NF,L)\}$.

In OGS, $1 \xrightarrow{V1O} 2 \xrightarrow{V2SC} 5$, and $3 \xrightarrow{V1O} 5$

Since $z_0 = \{1,3\}$, $\eta(z_0, \mathbf{V1O}) = \{2,5\}$ as explained below.

$x \in z_0$	$\eta(x,\mathbf{V1O})$	$UR(.)$	$UR(.)$ and $\lambda(x)$= (NF,NF,L)
1	$\eta(1,\mathbf{V1O})=2$	{2,5}	{2,5}
3	$\eta(3,\mathbf{V1O})=5$	{5}	{5}

Thus, $z_1 = \{2,5\}$.

The states of the components in OGS states 2 and 5 are given below.

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
2:	2	1	1	1	1	1	1	2	1	1
	O	C	C	OFF	NF	NF	L	O	(C,C)	(NF,OFF,NF)

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
5:	2	3	1	1	1	1	1	2	3	1
	O	SC	C	OFF	NF	NF	L	O	(SC,C)	(NF,OFF,NF)

After opening the cooling water, the event F1H occurs, and the flowmeter F1 reads high flow.

➔ **F1H, y_2 = (F,NF,L):** new event occurred with label change (Case-3)

Since $z_{k+1} = \xi_e(z_k, \sigma_{k+1}, y_{k+1})$, $\xi_e(z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_{k+1}\}$.

Then, $z_2=\{x \mid x \in UR(\eta(z_1, \mathbf{F1H}))$ and $\lambda(x)=\mathbf{(F,NF,L)}\}$

Since $z_1 = \{2,5\}$, $\eta(z_1, \mathbf{F1H}) = \{6,10\}$ as explained below.

$x \in z_1$	$\eta(x,$**F1H** $)$	$UR(.)$	$UR(.)$ and $\lambda(x)$=(F,NF,L)
2	$\eta(2,$**F1H** $)=6$	{6,10}	{6,10}
5	$\eta(5,$**F1H** $)=10$	{10}	{10}

Thus, $z_2 = \{6,10\}$

In states 6 and 10 of OGS, the states of components are as follows.

	V1	**V2**	**V3**	**PSU**	**F1**	**F2**	**AIT**	**INT1**	**INT2**	**INT3**
6:	2	1	1	1	2	1	1	2	1	3
	O	C	C	OFF	F	NF	L	O	(C,C)	F,OFF,NF

	V1	**V2**	**V3**	**PSU**	**F1**	**F2**	**AIT**	**INT1**	**INT2**	**INT3**
10:	2	3	1	1	2	1	1	2	3	3
	O	SC	C	OFF	F	NF	L	O	(SC,C)	F,OFF,NF

➔ **V2SC**

At this point, a stuck-closed failure happens in V2 that causes this valve to get closed. This event is unobservable and does not result in new observation at this point. The supervisor will then send the open command to valve 2 based on SUP1 logic (V2O command).

➔ **V2O , y_3 = (F,NF,L):** new event occurred in the true system with no label change (Case-1)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$

Then, $z_3 = \{x \mid x \in UR(\eta(z_2, \mathbf{V2O}))$ and $\lambda(x)=\mathbf{(F,NF,L)}\}$.

Since $z_2 = \{6,10\}$, $\eta(z_2, \mathbf{V2O}) = \{5,18\}$ as explained below.

$x \in z_2$	$\eta(x,$**V2O**$)$	***UR*(.)**	***UR*(.) and $\lambda(x)$=(F,NF,L)**
6	$\eta(6,$**V2O** $)=8$	{8}	{8}
10	$\eta(10,$**V2O** $)=15$	{15}	{15}

Thus, $z_3 = \{8,15\}$

In states 8 and 15 of OGS, the states of components are as follows.

	V1	**V2**	**V3**	**PSU**	**F1**	**F2**	**AIT**	**INT1**	**INT2**	**INT3**
8:	2	2	1	1	2	1	1	2	2	3
	O	O	C	OFF	F	NF	L	O	(O,C)	F,OFF,NF

	V1	**V2**	**V3**	**PSU**	**F1**	**F2**	**AIT**	**INT1**	**INT2**	**INT3**
15:	2	3	1	1	2	1	1	2	3	3
	O	SC	C	OFF	F	NF	L	O	(SC,C)	F,OFF,NF

Next, the supervisor opens valve 3 based on SUP1 logic by sending the V3O command

➔ **V3O ,** y_4**= (F,NF,L):** new event occurs in the true system with no label change (Case-1)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$

Then, $z_4 = \{x \mid x \in UR(\eta(z_3,$**V3O**$))$ and $\lambda(x)=$(F,NF,L)$\}$

Since $z_3 = \{8,15\}$, $\eta(z_3,$ **V3O**$) = \{12,21\}$ as explained below.

$x \in z_3$	$\eta(x,$**V3O** $)$	***UR*(.)**	***UR*(.) and $\lambda(x)$=(F,NF,L)**
8	$\eta(8,$**V3O** $)=12$	{12}	{12}
15	$\eta(15,$**V3O** $)=21$	{21}	{21}

Thus, $z_4 = \{12,21\}$

In states 12 and 21 of OGS, the states of components are as follows.

	V1	**V2**	**V3**	**PSU**	**F1**	**F2**	**AIT**	**INT1**	**INT2**	**INT3**
12:	2	2	2	1	2	1	1	2	5	3
	O	O	O	OFF	F	NF	L	O	(O,O)	F,OFF,NF

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
21:	2	3	2	1	2	1	1	2	6	3
	O	SC	O	OFF	F	NF	L	O	(SC,O)	F,OFF,NF

Since V2 was commanded to open but it is stuck-closed and V3 is opened, the oxygen flow will not build up causing no change in the label. Then, the supervisor starts the power supply unit.

➔ **Run**, y_5 **= (F,NF,L)** new event occurs at the true system with no label change (Case-1)

Since $z_{k+1} = \xi_\sigma(z_k, \sigma_{k+1})$, $\xi_\sigma(z_k, \sigma_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_k\}$.

Then, $z_5 = \{x \mid x \in UR(\eta(z_4,$ **Run**$))$ and $\lambda(x)=$(F,NF,L)$\}$

Since $z_4 = \{12,21\}$, $\eta(z_4,$ **Run**$) = \{17,27\}$ as explained below.

$x \in z_3$	$\eta(x,$V3O)	UR(.)	UR(.) and $\lambda(x)$=(F,NF,L)
12	η(12,**Run**)=17	{17}	{17}
21	η(21,**Run**)=27	{27}	{27}

Thus, $z_5 = \{17,27\}$

In states 17 and 27 of OGS, the states of components are as follows.

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
17:	2	2	2	2	2	1	1	2	5	5
	O	O	O	ON	F	NF	L	O	(O,O)	F,ON,NF

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
27:	2	3	2	2	2	1	1	2	6	5
	O	SC	O	ON	F	NF	L	O	(SC,O)	F,ON,NF

Once the power supply unit is started, the oxygen that is already existing inside the vessel will be transformed to ozone causing a high reading of the ozone concentration meter.

➔ **O3H**, y_6 **= (F,NF,H):** new event occurred with label change (Case-3)

Since $z_{k+1} = \xi_e(z_k, \sigma_{k+1}, y_{k+1})$, $\xi_e(z_k, \sigma_{k+1}, y_{k+1}) = \{x \mid x \in UR(\eta(z_k, \sigma_{k+1}))$ and $\lambda(x)=y_{k+1}\}$

Then, $z_6 = \{x \mid x \in UR(\eta(z_5,$ **O3H**$))$ and $\lambda(x)=$(**F,NF,H**)$\}$

To get $\eta(z_5, \mathbf{O3H})$, since z_5= {17,27}

Since z_5 = {17,27}, and O3H is not defined in either state, z_6= {17,27}= ϕ as explained below.

$x \in z_5$	$\eta(x,$F1H)	$UR(.)$	$UR(.)$ and $\lambda(x)$=(F,NF,H)
17	η(17,**O3H**)= ---	---	---
27	η(27,**O3H**)= ---	---	---

DISCREPANCY DETECTED. This is shown in Figure 5.8.

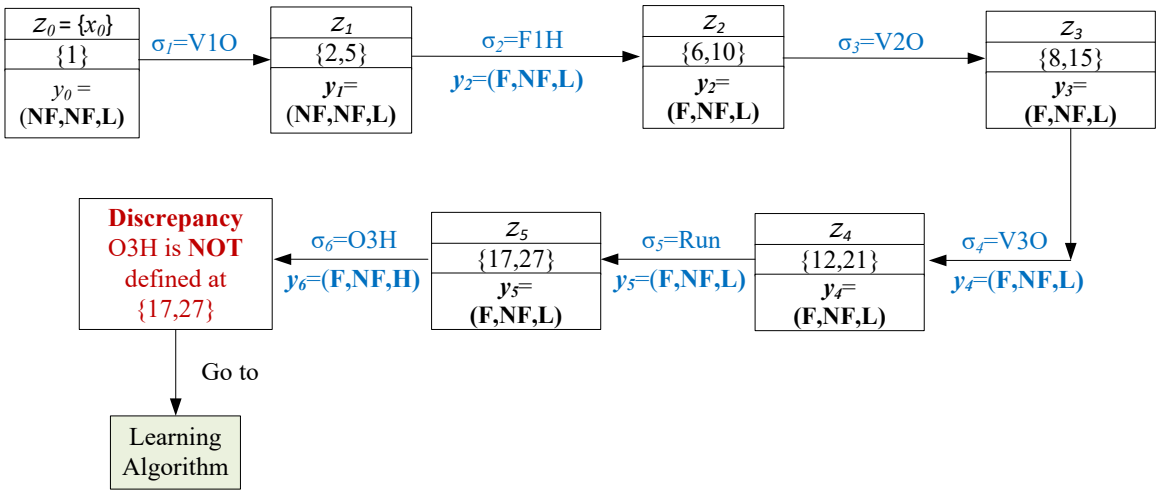


Figure 5.8: Ozone System Progression and Discrepancy Detection

5.4 The Learning Algorithm

The learning algorithm of missing transitions with observable events is explained in section 4.5.3

1- **STEP1: Detect the Discrepancy and Determine the Discrepancy Type.**

   a. **Detect discrepancy and its Type**

      i. **Type of Discrepancy**: ***Unexpected Observable Event***.

      ii. **Cause: Missing Transition with Observable Event**

   b. **Collect the Current System Observations and Information**

      i. Event of discrepancy σ_d= **O3H**

      ii. Actual Label of Discrepancy **ALD**= y_d= **(F,NF,H)**

      iii. Last Conforming State **LCS:** z_L= z_5 = {17,27}

2- STEP-2: Analyzing the Collected Information and Concluding Sources and Candidates

a. Suspected Components:

Since $\boldsymbol{SC} = \{C \mid \sigma_d \in \Sigma_c\}$ and σ_d=O3H, and O3H is an event of{AIT, INT3,SUP2}. Thus,

SC = {AIT, INT3,SUP2}

At x_{L1}= 17

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
17:	2	2	2	2	2	1	1	2	5	5
	O	O	O	ON	F	NF	L	O	(O,O)	F,ON,NF

O3H is defined at AIT: L

O3H is **NOT** defined at INT3: (F,ON,NF)

O3H is defined at SUP2:L

Then, by checking x_{L1}= 17, the deficient suspected components $\mathbf{DSC}_1$ ={INT3}

And the complete suspected components CSC_1={AIT,SUP2}

The irrelevant components $Cirv_1$ = All components – {AIT, INT3,SUP2}

At x_{L2}= 27 where,

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
27:	2	3	2	2	2	1	1	2	6	5
	O	SC	O	ON	F	NF	L	O	(SC,O)	F,ON,NF

O3H is defined at AIT:L

O3H is **NOT** defined at INT3: (F,ON,NF)

O3H is defined at SUP2:L

Then, by checking x_{L2}= 27, the deficient suspected components $\mathbf{DSC}_2$ ={INT3}

And the complete suspected components CSC_2={AIT,SUP2}

The irrelevant components $Cirv_2$= All components – {AIT, INT3,SUP2}

Deficient Components Table

x_{Lj}	DSC_j	CSC_j	$Cirv_j$
x_{L1}=17	{INT3}	CSC_1={AIT,SUP2}	C - {AIT,SUP2}
x_{L2}=27	{INT3}	CSC_2={AIT,SUP2}	C - {AIT,SUP2}

Note: for readability, the deficient components table here is shorter version of the ones introduced in the mathematical formulation section 4.5.3, as we omitted $x_{Lj,\{DSC\}}$, $x_{Lj,\{CSC\}}$, and $x_{Lj,\{Cirv\}}$.

b. <u>Determine the Candidate Suspected States</u>

<u>Proposed States X_{PS}</u>

Since $X_{PS}=\{x \mid \lambda(x)=ALD\}$, Thus $X_{PS}=\{x \mid \lambda(x)=$**(F,NF,H)**$\}$

These states are obtained from the output map by finding the states that can generate the same label. It should be noted that the states in the output map table (i.e., the domain of the map) are generated by the cartesian product of the components' state sets. In this case study, the output map table has 18432 rows. Furthermore, it should be noted that the output label is evaluated based on the corresponding values of F1, F2 and AIT. So, this step will get all the states in the output map table that have a label of **(F,NF,H).**

This step returns 2304 rows from the output map table which is a large number of states that is hard to list them here. This is because in this step, the proposed states are not yet filtered by the rules that evaluate the candidate expected states. Thus, for readability, we will provide the list of filtered proposed states in the next step.

<u>Candidate expected states</u>

The candidate expected states could be obtained by filtering the proposed states above using the rules listed in section 4.5.3 of the algorithm of missing observable events. Basically, the state of relevant components is updated with the new event. Thus, AIT after O3H should be at state H. The "irrelevant" components stay at the same state as their state never change with the new event O3H. Thus, all other components should stay at their same state before the event O3H occurred.

	V1	**V2**	**V3**	**PSU**	**F1**	**F2**	**AIT**	**INT1**	**INT2**	**INT3**
17:	2	2	2	2	2	1	1	2	5	5
	O	O	O	ON	F	NF	L	O	(O,O)	F,ON,NF

Table 5.1 lists the candidate states from the state 17.

Table 5.1: Ozone Generation Case Study – Filtered Proposed States from State 17

V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3	Label
O	O	O	ON	F	NF	H	O	(O,O)	(NF,**OFF**,NF)	F,NF,H
O	O	O	ON	F	NF	H	O	(O,O)	(NF,**OFF**,F)	F,NF,H
O	O	O	ON	F	NF	H	O	(O,O)	(**NF**,ON,NF)	F,NF,H
O	O	O	ON	F	NF	H	O	(O,O)	(NF,ON,**F**)	F,NF,H
O	O	O	ON	F	NF	H	O	(O,O)	(F,**OFF**,NF)	F,NF,H
O	O	O	ON	F	NF	H	O	(O,O)	(F,**OFF**,F)	F,NF,H
O	O	O	ON	F	NF	H	O	(O,O)	**(F,ON,NF)**	F,NF,H
O	O	O	ON	F	NF	H	O	(O,O)	(F,ON,**F**)	F,NF,H

Table 5.2 lists the candidate states from the state 27.

	V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3
27:	2	3	2	2	2	1	1	2	6	5
	O	SC	O	ON	F	NF	L	O	(SC,O)	F,ON,NF

Table 5.2: Ozone Generation Case Study – Filtered Proposed States from State 17

V1	V2	V3	PSU	F1	F2	AIT	INT1	INT2	INT3	Label
O	SC	O	ON	F	NF	H	O	(SC,O)	(NF,**OFF**,NF)	F,NF,H
O	SC	O	ON	F	NF	H	O	(SC,O)	(NF,**OFF**,F)	F,NF,H
O	SC	O	ON	F	NF	H	O	(SC,O)	(**NF**,ON,NF)	F,NF,H
O	SC	O	ON	F	NF	H	O	(SC,O)	(NF,ON,**F**)	F,NF,H
O	SC	O	ON	F	NF	H	O	(SC,O)	(F,**OFF**,NF)	F,NF,H
O	SC	O	ON	F	NF	H	O	(SC,O)	(F,**OFF**,F)	F,NF,H
O	SC	O	ON	F	NF	H	O	(SC,O)	**(F,ON,NF)**	F,NF,H
O	SC	O	ON	F	NF	H	O	(SC,O)	(F,ON,**F**)	F,NF,H

Since the interaction INT3 = AIT_F1_PSU_F2 is formed by the synchronous product of F1,PSU,F2, then for each state in

Table 5.1 and Table 5.2 above of filtered proposed state, this interaction should follow the same states of its comprising components. This means that, since components F1 reached the state F, and PSU reached ON and F2 reached NF, then INT3 state should be at (F,ON,NF). Consequently, the candidate expected state has (F,ON,NF) in its INT3 while all the other states will be rejected because they are just produced

from the cartesian product, but they cannot be reached in this case.

Candidate expected states

x_{Lj}	$X_{cndd}(x_{Lj})$
x_{L1}=17	(O,O,O,ON,F,NF,H,O,(O,O),(F,ON,NF))
x_{L2}=27	(O,SC,O,ON,F,NF,H,O,(O,O),(F,ON,NF))

c. Determine the Destinations of the Missing Transitions

Destinations Table:

$Dest_{j,k,i} = Ci: x_{cndd\_k,i}(x_{Lj})$ if $Ci \in DSC_{jk} \forall\ 1 \le j \le |Z_L=LCS|, \forall\ 1 \le k \le |Xcndd(x_{Lj})|, \forall\ 1 \le i \le |C|$

Table 5.3: Ozone Generation Case Study- Destinations Table

x_{Lj}	$x_{cndd\_k}(x_{Lj})$	**INT3**
x_{L1}=17	$x_{cndd\_1}(x_{L1})$=(O,O,O,ON,F,NF,H,O,(O,O),(F,ON,NF))	(F,ON,NF)
x_{L2}=12	$x_{cndd\_1}(x_{L2})$=(O,SC,O,ON,F,NF,H,O,(O,O),(F,ON,NF))	(F,ON,NF)

Note: The columns of all the components are omitted from the destination table for simplicity; INT3 is the 10th component.

d. Determine the Sources of Missing Transitions

Sources Table

$Srcs(Dest_{j,k,i}) = C_i: x_{Lj,i}$ if $C_i \in DSC_{j,k}$ $\forall\ 1 \le j \le |Z_L=LCS|, \forall\ 1 \le k \le |Xcndd(x_{Lj})|, \forall\ 1 \le i \le |C|$

Table 5.4: Ozone Generation Case Study- Sources Table

x_{Lj}	$x_{cndd\_k}(x_{Lj})$	**INT3**
x_{L1}=17	$x_{cndd\_1}(x_{L1})$=(O,O,O,ON,F,NF,L,O,(O,O),(F,ON,NF))	(F,ON,NF)
x_{L2}=27	$x_{cndd\_1}(x_{L2})$=(O,SC,O,ON,F,NF,L,O,(SC,O),(F,ON,NF))	(F,ON,NF)

Note: The columns of all the components are omitted from the Sources table for simplicity where INT3 is the 10th component.

3- STEP-3: Hypotheses Generation and Model Compliance Verification

Hypotheses:

$H_{j,k,i} = Srcs(Dest_{j,k,i}) \xrightarrow{\sigma_d} Dest_{j,k,i} \; \forall \; 1 \leq j \leq |z_L = LCS|, \; \forall \; 1 \leq k \leq |Xcndd(x_{Lj})|, \; \forall \; 1 \leq i \leq |C|$

$H_{1,1,10} = \; INT3{:}(F,ON,NF) \xrightarrow{O3H} (F,ON,NF)$

As shown, the algorithm was able to detect the missing transition and a hypothesis based on it.

5.5 Remarks

This case study could be used to drive a new rule in the algorithm that is to speed up the calculations rather than going through all the steps of the algorithm. Based on the nature of constructing the automata of interactions, usually the events of the affected components are used to add self-loop transitions at the affecting components. For example, the flowmeter F1 was an affected component by the state of the affecting component which is valve V1. Therefore, the interaction INT1 was built by self-looping F1 event at the states of V1. Using this idea, we can drive a new rule that when the missing transition is found in an interaction automaton AND the event of discrepancy belongs to the affected component, it would be possible to just self-loop this event at the last conforming state of the interaction automaton. This is because as was seen in this case study, the source state was the same as destination state while the missing transition was a self-loop.

Chapter 6

Accomplishments and Contributions

After introducing the problem of missing transitions and its impact on fault diagnosis, this study continued with formulating the solution as a learning algorithm that generates hypotheses for adding the missing transition to the individual components' models. As mentioned in the literature review in chapter 1, the approach in [3] can be considered as the launching point of this thesis. In [3], the authors started with an incomplete model of a system (nominal model) that misses some transitions which lead to discrepancy between the observations and the nominal model. Then, the missing transitions are hypothesized to be added to the flat model of the entire system. As the possible hypotheses could be large, the Parsimonious Covering Theory (PCT) is used to narrow down the list of hypotheses. In this section, the proposed method in this thesis and the approach in [3] are compared.

Suppose we have a system comprised of two components, represented by automata G1 and G2 shown in Figure 6.1.

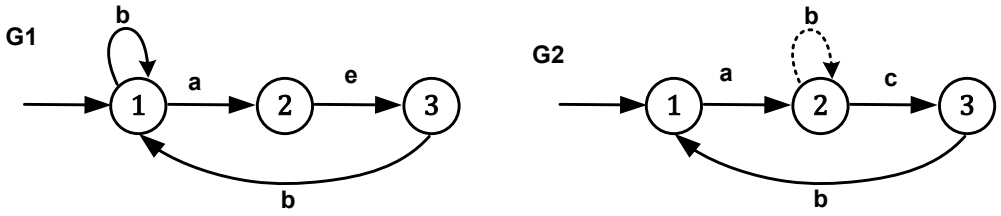


Figure 6.1: Accomplishments Discussion Example Automata

The true model G is built by the synchronous product of the two automata.

G = sync (**G1**, **G2**)

The resulting automaton is the true model that has 7 states and 10 transitions, as shown in Figure 6.2.

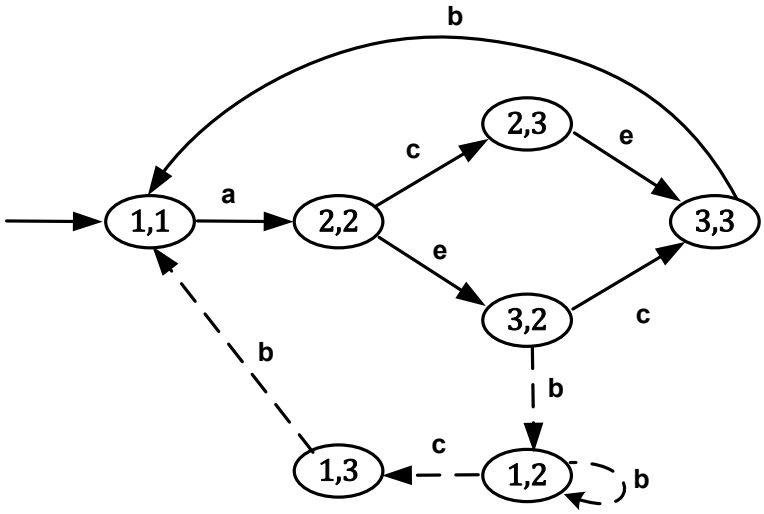


Figure 6.2: Accomplishments Discussion Example True model

Note that in Figure 6.2, the dotted lines do not represent unobservable events as in the previous chapters, and they represent the part that will be missing if event b at state 2 in G2 is missing as explained in the following.

Let us assume that, due to human mistake or lack of knowledge in the nominal model, the self-loop transition of event b at state 2 in automaton G2 is missing, (the dotted line). After finding the synchronous product of the two components with $2 \xrightarrow{b} 2$ removed from G2, we obtain the nominal model that has 5 states and 6 transitions as shown in Figure 6.3.

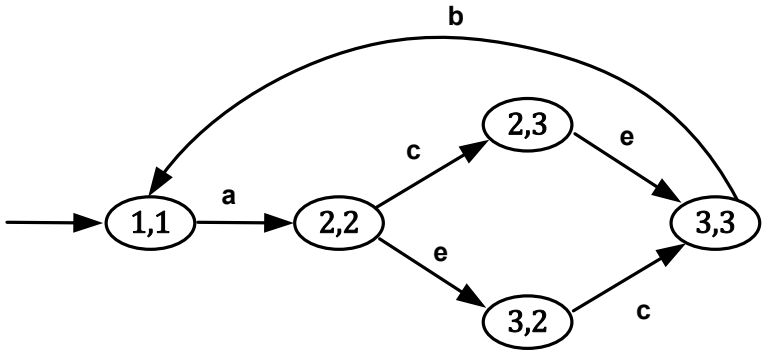


Figure 6.3: Accomplishments Discussion Example Nominal Model

By comparing the true model and the nominal model, it is obvious that the nominal model is surprisingly smaller than the true model in terms of the number of states and consequently has less transitions.

Hence, the approach tackled in [3] to hypothesize missing transitions on top of an entire nominal model of a system (Like the in the example) might not be correct because the nominal models could be missing some states. In addition, any added transitions still will be unable to correct this nominal model. In contrast, this thesis works on correcting the problem on the level of the components (G2 in the example) to make sure that hypothesizing the missing transition will correct the number of states of the model and the added transitions will pass across their corresponding states. Another benefit is that working on the level of components and after the synchronous product, the new transitions will be propagated to many places in the nominal model. This means that, a newly added transition in a component can fix many missing transitions in the nominal model at once rather than waiting to detect their discrepancy. This is shown in the example, by assuming that we worked on recovering the missing transition in G2 and we rebuilt the nominal model. Then, adding a single transition in G2 will result in adding four transitions in the nominal model.

Chapter 7

Conclusion

7.1 Summary

This research tackles the problem of having incomplete DES models. The main goal of the research was to come up with a learning algorithm that generates hypotheses for the missing transitions. The study started with operating a set of experiments taking into consideration the different types of events in the model transitions. Observable, unobservable, private and common events were addressed in the experiments. This was followed by formal discussion that resulted in learning rules for amending the DES model to account for discrepancies.

Unlike the learning methods that rely on the flat model of the plant, the proposed method takes advantage of the structure of the plant and models inaccuracies of models as missing transitions in component models. This leads to more realistic hypotheses and amend the overall plant models at several states (rather than one).

Since the algorithm is to be incorporated with real systems that have unobservable events occurring during the operation, an event-state observer was introduced to monitor the model progression according to the occurred events and generated output labels. The observer in the proposed scheme, was used to detect the discrepancy in the model if the model behavior mismatches the real system. This discrepancy is forwarded to the learning algorithm to work on it. The learning algorithm in turn collects the currently existing information from the real system and the nominal model, then by analyzing this information, it determines the suspected components and the sources and destinations of the missing transitions. Finally, the algorithm generates the hypotheses for the missing transitions. The new hypotheses will be used to generate new models where these models be examined using new observation to see their compliance. The models that are found incompliant will be rejected while the compliant ones will be kept.

7.2 Scope of Applicability

The scope of applicability for the learning algorithm introduced in this research defines how far this algorithm can be applicable; in other words, the range of DES systems that can use this algorithm. The

scope is governed by the limitations imposed by the assumptions made in this thesis. Based on the assumptions made in this research, it is possible for a DES to use the learning algorithm if the following conditions hold.

1. The system is modeled as a deterministic Moore automaton.
2. The fault modes (abnormal states) in the components are permanent such that if the fault event happens, the component state does not return to normal.
3. The components are modeled as automata with correct number of states but perhaps missing some transitions between the states.
4. The number of components is exactly known, and there are no missing components, and there are no new components to be added by the algorithm.
5. The event set for each component is known. This means that no new events (symbols) need to be added to the event sets.
6. The set of output labels is known. Also, the output map is defined and known in terms of the states and outputs.
7. The output map function of the true system and the nominal system are the same.
8. If the missing transition has an unobservable event, this event must be diagnosable to able to use the algorithm to hypothesize the missing transition.

7.3 Future Work

This research tackled the untimed version of automata. Since, in the real applications including the timing information enriches the accuracy of models, future research can extend the work to the learning of missing transitions in timed automata. In addition, this approach addressed the problem of missing transitions while assuming the correct number of states for the components. Future research can work on the case of discovering missing states in the components.

In terms of the learning technique, the tabulation of information during determining sources, destination candidates as well as the successful and rejected hypotheses will create a rich foundation to integrate this work with an artificial intelligence technique such as Reinforcement Learning.

Furthermore, this approach introduces a more precise approach to hypothesizing missing transitions compared to the approach in [3] that could be prone to huge number of hypotheses. Therefore, it is an open direction of research to integrate our approach of creating hypotheses in this thesis with the learning diagnoser in [3].

References

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. 3rd ed. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-72274-6.

[2] W. Murray Wonham and Kai Cai, *Supervisory Control of Discrete-Event Systems*. Springer, 2019. https://link.springer.com/book/10.1007/978-3-319-77452-7

[3] R. H. Kwong and D. L. Yonge-Mallo, "Fault diagnosis in discrete-event systems: Incomplete models and learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 118–130, Feb. 2011, doi: 10.1109/TSMCB.2010.2047257.

[4] J. Zhang, J. Cao, and D. Wu, "On the transformation from automata into Petri Nets in FMS," in *2010 IEEE International Conference on Mechatronics and Automation*, Aug. 2010, pp. 1647–1651. doi: 10.1109/ICMA.2010.5588899.

[5] M. Thirumaran, P. Dhavachelvan, D. Aishwarya, and K. Rajakumari, "Conventional Usage of Finite State Machine over Petri Net inWeb Service Change Management Framework," *IERI Procedia*, vol. 4, pp. 99–109, Jan. 2013, doi: 10.1016/j.ieri.2013.11.016.

[6] T. Lu and X. Huo, "Modeling and verification of supervisory rules in Internet of Things environment," *Procedia Computer Science*, vol. 126, pp. 1926–1935, Jan. 2018, doi: 10.1016/j.procs.2018.08.062.

[7] J. Li, D. Lefebvre, C. N. Hadjicostis, and Z. Li, "Observers for a class of timed automata based on elapsed time graphs," *IEEE Transactions on Automatic Control*, vol. 67, no. 2, pp. 767–779, Feb. 2022, doi: 10.1109/TAC.2021.3064542.

[8] I. Tahiri, A. Philippot, V. Carré-Ménétrier, and A. Tajer, "Timed synthesis control approach for tolerant-fault control of discrete event systems (DES)," in *2018 International Conference on Control, Automation and Diagnosis (ICCAD)*, Mar. 2018, pp. 1–6. doi: 10.1109/CADIAG.2018.8751490.

[9] N. Bertrand, P. Bouyer, T. Brihaye, Q. Menete, C. Baier, M. Grosser and M. Jurdzinski, "Stochastic Timed Automata," *Log.Meth.Comput.Sci.*, vol. 10, no. 4, p. 6, Dec. 2014, doi: 10.2168/LMCS-10(4:6)2014.

[10] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," *Computer Aided Verification*, vol. 6806, G. Gopalakrishnan and S. Qadeer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 585–591. doi: 10.1007/978-3-642-22110-1_47.

[11] J. N. Mordeson and D. S. Malik, *Fuzzy Automata and Languages: Theory and Applications*, 1st ed. New York: Chapman and Hall/CRC, 2002. doi: 10.1201/9781420035643.

[12] I. Micić, Z. Jančić, J. Ignjatović, and M. Ćirić, "Determinization of fuzzy automata by means of the degrees of language inclusion," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2144–2153, Dec. 2015, doi: 10.1109/TFUZZ.2015.2404348.

[13] I. Wendell Bates, A. Karimoddini, and M. Karimadini, "Learning a partially-known discrete event system," *IEEE Access*, vol. 8, pp. 61806–61816, 2020, doi: 10.1109/ACCESS.2020.2983074.

[14] R. K. Singh, A. Rani, and M. K. Sachan, "Fuzzy automata: a quantitative review," *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 3, no. 7, Art. no. 7, Jul. 2017.

[15] A. M. Idghamishi and S. Hashtrudi Zad, "Fault diagnosis in hierarchical discrete-event systems," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, Dec. 2004, vol. 1, pp. 63-68 Vol.1. doi: 10.1109/CDC.2004.1428607.

[16] A. Mohammadi-Idghamishi and S. Hashtrudi-Zad, "Hierarchical fault diagnosis: application to an ozone plant," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 5, pp. 1040–1047, Sep. 2007, doi: 10.1109/TSMCC.2007.900622.

[17] X. Zhao and D. Ouyang, "On-line diagnosis of discrete-event systems: a hierarchical approach," in *2008 IEEE Conference on Robotics, Automation and Mechatronics*, Sep. 2008, pp. 785–790. doi: 10.1109/RAMECH.2008.4681372.

[18] S. Reshmila and R. Devanathan, "Diagnosis of power system failures using observer based discrete event system," in *2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI)*, Jan. 2016, pp. 131–135. doi: 10.1109/CMI.2016.7413725.

[19] A. White, A. Karimoddini, and R. Su, "Fault diagnosis of discrete event systems under unknown initial conditions," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 5246–5252, Dec. 2019, doi: 10.1109/TAC.2019.2912712.

[20] S. Hashtrudi Zad, R. H. Kwong, and W. M. Wonham, "Fault diagnosis in discrete-event systems: framework and model reduction," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1199–1212, Jul. 2003, doi: 10.1109/TAC.2003.814099.

[21] H. Hu, H. Li, Y. Jiang, Y. Zheng, S. Huang, and Y.-F. Sheng, "Fault diagnosis based on discrete event system for power grid," *The 27th Chinese Control and Decision Conference (2015 CCDC)*, May 2015, pp. 2668–2672. doi: 10.1109/CCDC.2015.7162383.

[22] A. Ghasaei, Z. J. Zhang, W. M. Wonham, and R. Iravani, "A discrete-event supervisory control for the AC microgrid," *IEEE Transactions on Power Delivery*, vol. 36, no. 2, pp. 663–675, Apr. 2021, doi: 10.1109/TPWRD.2020.2988687.

[23] F. E. H. Tay and L. Shen, "Fault diagnosis based on Rough Set Theory," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 1, pp. 39–43, Feb. 2003, doi: 10.1016/S0952-1976(03)00022-8.

[24] X. Yunfang, H. Limin, F. Xinqiao, and L. Weina, "Application of Rough Set Theory in the fault diagnosis of distribution line," in *2007 8th International Conference on Electronic Measurement and Instruments*, Aug. 2007, pp. 3-731-3–736. doi: 10.1109/ICEMI.2007.4351021.

[25] X. Geng, D. Ouyang, and Y. Zhang, "Model-based diagnosis of incomplete discrete-event system with rough set theory," *Sci. China Inf. Sci.*, vol. 60, no. 1, p. 012205, Nov. 2016, doi: 10.1007/s11432-015-0897-0.

[26] X. Wang, "Research on network fault diagnosis based on fault tree analysis," in 2022 IEEE Asia-*Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, Apr. 2022, pp. 1186–1189. doi: 10.1109/IPEC54454.2022.9777312.

[27] R. H. Kwong and D. L. Yonge-Mallo, "Fault Diagnosis in Discrete-Event Systems with incomplete models: learnability and diagnosability," *IEEE Transactions on Cybernetics*, vol. 45, no. 7, pp. 1236–1249, Jul. 2015, doi: 10.1109/TCYB.2014.2347801.

[28] M. Sayed-Mouchaweh, A. Philippot, V. Carré-Ménétrier, and B. Riera, "Timed-event-state-based diagnoser for manufacturing systems," in *Information Technology For Balanced Manufacturing Systems*, Boston, MA, 2006, pp. 415–424. doi: 10.1007/978-0-387-36594-7_44.

[29] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annual Reviews in Control*, vol. 37, no. 2, pp. 308–320, Dec. 2013, doi: 10.1016/j.arcontrol.2013.09.009.

[30] X. Yin, J. Chen, Z. Li, and S. Li, "Robust fault diagnosis of stochastic discrete event systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4237–4244, Oct. 2019, doi: 10.1109/TAC.2019.2893873.

[31] X. Yin and S. Li, “Robust diagnosability and robust prognosability of discrete-event systems revisited,” in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Jul. 2018, pp. 302–307.doi: 10.1109/CYBER.2018.8688259.

[32] F. G. Cabral and M. V. Moreira, “Synchronous diagnosis of discrete-event systems,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 921–932, Apr. 2020, doi: 10.1109/TASE.2019.2951627.

[33] Z. Long-Mei, L. Wei, and W. Jing, “Partial diagnosability analysis of discrete event systems,” in *2016 International Symposium on Computer, Consumer and Control (IS3C)*, Jul. 2016, pp. 128–131. doi: 10.1109/IS3C.2016.43.

[34] A. P. Estrada-Vargas, E. López-Mellado, and J.-J. Lesage, “A black-box identification method for automated discrete-event systems,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 3, pp. 1321–1336, Jul. 2017, doi: 10.1109/TASE.2015.2445332.

[35] F. Basile and L. Ferrara, “Identification of timed input/output relationships for industrial automation systems using timed interpreted Petri Nets,” in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Oct. 2019, pp. 1831–1837. doi: 10.1109/SMC.2019.8914396.

[36] J. G. V. de Castro, G. S. Viana, and M. V. Moreira, “Distributed identification of discrete-event systems with the aim of fault detection,” IFAC-PapersOnLine, vol. 55, no. 28, pp. 309–314, Jan. 2022, doi: 10.1016/j.ifacol.2022.10.359.

[37] M. V. Moreira and J.-J. Lesage, “Discrete event system identification with the aim of fault detection,” *Discrete Event Dyn Syst*, vol. 29, no. 2, pp. 191–209, Jun. 2019, doi: 10.1007/s10626-019-00283-z.

[38] M. V. Moreira and J.-J. Lesage, “Fault diagnosis based on identified discrete-event models,” *Control Engineering Practice*, vol. 91, p. 104101, Oct. 2019, doi: 10.1016/j.conengprac.2019.07.019.

[39] R. P. C. de Souza, M. V. Moreira, and J.-J. Lesage, “A timed model for discrete event system identification and fault detection,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 808–813, Jan. 2020, doi: 10.1016/j.ifacol.2020.12.835.

[40] R. P. C. de Souza, M. V. Moreira, and J.-J. Lesage, “A hierarchical approach for discrete-event model identification incorporating expert knowledge,” IFAC-PapersOnLine, vol. 53, no. 4, pp. 275–281, Jan. 2020, doi: 10.1016/j.ifacol.2021.04.065.

[41] Christopher M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006. https://link.springer.com/book/9780387310732.

[42] E. Alpaydin, Introduction to Machine Learning, 4th ed. Massachusetts: The MIT Press, 2020. Available: https://mitpress.mit.edu/9780262043793/introduction-to-machine-learning/

[43] A. Maier, “Online passive learning of timed automata for cyber-physical production systems,” Jul. 2014. doi: 10.1109/INDIN.2014.6945484.

[44] J. An, L. Wang, B. Zhan, N. Zhan, and M. Zhang, “Learning real-time automata,” *Sci. China Inf. Sci.*, vol. 64, no. 9, p. 192103, Aug. 2021, doi: 10.1007/s11432-019-2767-4.

[45] B. Wu, X. Zhang, and H. Lin, “Permissive supervisor synthesis for Markov Decision Processes through learning,” *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3332–3338, Aug. 2019, doi: 10.1109/TAC.2018.2879505.

[46] Huimin Zhang, Lei Feng, and Zhiwu Li, “A learning-based synthesis approach to the supremal nonblocking supervisor of discrete-event systems,” *IEEE Transactions on Automatic Control*, vol. 63, no. 10, pp. 3345–3360, Oct. 2018, doi: 10.1109/TAC.2018.2793662.

[47] I. W. Bates, A. Karimoddini, and M. Karimadini, "A learning-based Approach for diagnosis and diagnosability of unknown discrete event systems," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2022, doi: 10.1109/TNNLS.2022.3204557.

[48] Y. Peng and J. A. Reggia, *Abductive Inference Models for Diagnostic Problem-Solving*. New York: Springer-Verlag, 1990.

[49] G. Giantamidis and S. Tripakis, "Learning Moore machines from input-output traces," in *FM 2016: Formal Methods*, Cham, 2016, pp. 291–309. doi: 10.1007/978-3-319-48989-6_18.

[50] S. H. Zad, Shauheen Zahirazami, Farzam Boroomand, *"Discrete Event Control Kit (DECK 1.2013.11),"* https://users.encs.concordia.ca/~shz/deck/

[51] A. Hélias, F. Guerrin, and J. Steyer, "Abstraction of continuous system trajectories into timed automata," *IFAC Proceedings Volumes*, vol. 37, no. 18, pp. 309–314, Sep. 2004, doi: 10.1016/S1474-6670(17)30764-4.

[52] E. Francis, *Ozone in Water and Wastewater Treatment*, 3rd ed. Michigan: Ann Arbor Science Publishers, 1972.

[53] M. Sipser, *Introduction to the theory of computation*, Third edition. Boston, MA: Cengage Learning, 2013.

[54] R. Sedgewick, *Algorithms in Java, Third Edition, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching*, 3rd ed. Boston, MA: Addison-Wesley Professional, 2002. Available: https://www.oreilly.com/library/view/algorithms-in-java/0201361205/